DEVELOPMENT OF AN INTEGRATED INTERFACE MODELLING METHODOLOGY TO SUPPORT SYSTEM ARCHITECTURE ANALYSIS

Amad UDDIN

Submitted for the Degree of

Doctor of Philosophy

Faculty of Engineering and Informatics

University of Bradford

2016

Abstract

Amad Uddin

Development of an Integrated Interface Modelling Methodology to Support System Architecture Analysis

Keywords: Systems engineering, Engineering design, Interface, Interaction, System architecture, System-of-systems, and Design methodology

This thesis presents the development and validation of a novel interface modelling methodology integrated with a system architectural analysis framework that emphasises the need to manage the integrity of deriving and allocating requirements across multiple levels of abstraction in a structured manner.

The state of the art review in this research shows that there is no shared or complete interface definition model that could integrate diverse interaction viewpoints for defining system requirements with complete information. Furthermore, while existing system modelling approaches define system architecture with functions and their allocation to subsystems to meet system requirements, they do not robustly address the importance of considering well-defined interfaces in an integrated manner at each level of systems hierarchy. This results in decomposition and integration issues across the multiple levels of systems hierarchy. Therefore, this thesis develops and validates following:

- Interface Analysis Template as a systematic tool that integrates diverse interaction viewpoints for modelling system interfaces with intensive information for deriving requirements.
- Coupling Matrix as an architecture analysis framework that not only allocates functions to subsystems to meet requirements but also promotes consistent consideration of well-defined interfaces at each level of design hierarchy.

Insights from the validation of developed approach with engineering case studies within an automotive OEM are discussed, reflecting on the effectiveness, efficiency and usability of the methods.

Acknowledgement

This thesis would not have been possible without the support of many people. I would like to thank sincerely my supervisor Prof. Felician Campean for supervising me and giving me an opportunity to become a part of his research team. I would also like to thank my second supervisor Prof. M. Khurshid Khan for his constant support in this work.

Many thanks to members of an automotive company for their valuable input in providing case studies that really helped in testing the research ideas. Thanks to COMSATS Institute of Information Technology (CIIT) in allowing me to pursue my PhD with constant support throughout.

I would like to thank my research colleagues Dr. Unal Yildirim, Dr. Mohammed Reza Kianifar, and Dr. Guilhermina Torrao at Automotive Research Centre. Moments of listening to their research problems and sharing mine with them over lunch and tea times have been great learning opportunities.

This thesis is dedicated to my dear family members who stood by me at all times throughout this journey. My parents' and siblings' unconditional support and prayers helped me doing what I wanted to do and I don't have words to thank my father Aftab, my mother Mussarat, my sisters Maliha, Nadia, and my brother Sabih. I would like to specially thank the most amazing person, my wife Maryam (a PhD, best friend and life partner) who has been a truly inspiring, and encouraging character in my research journey. In the end, I would also thank Parivash (my two year old child), the most beautiful daughter who brought priceless joy in my life. She made me even more emotional and responsible person. Thanks to all these loving people from the core of my heart. This journey would not have been possible without your love, patience, encouragement, and support.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

BEQIC	Bradford Engineering Quality Improvement Centre
CM	Coupling Matrix
IAT	Interface Analysis Template
ICD	Interface Control Document
IRD	Interface Requirement Document
IRS	Interface Requirement Specification
FMA	Failure Mode Avoidance

1. Introduction and Objectives

1.1 Background

In today's global competitive design and manufacturing world, there is an unceasing pressure on companies to design complex yet cost effective technical systems of higher quality in less time by accomplishing the changing customer demands (Da Silveira et al., 2001; Holman et al., 2003). The rapidly changing customer demands have forced the companies to develop highly interactive technical systems by shifting and enhancing their product development process from monodisciplinary to multidisciplinary working environment. This is due to the underlying fact that today's modern engineered systems (such as automotive systems e.g. automobile) in turn constitute diverse and independent but integrated technological systems (e.g. gasoline engine, electric motor, solar panel, and steam engine) embedded with finite multidisciplinary features (i.e. electrical, mechanical, and software & control related) that work together to deliver expected requirements and functions articulated by designers for a whole system (e.g. hybrid vehicle). Designers from different disciplines (i.e. electrical, mechanical, and software & control) generate technical descriptions of the systems by analysing them via different views and via various sets of tools throughout the product development process. This shift of increasing multi-disciplinarity of technical systems due to involvement of multiple stakeholders, and achievement of multiple goals simultaneously has caused many inter-disciplinary or inter-operability problems to companies (Tomiyama, 2012).

In order to tackle the inter-operability problems with a strong focus on technical system's decomposition and integration issues with operational, functional, and physical models within systems hierarchy, systems engineering seems to be a promising solution at hand. "Systems engineering is an interdisciplinary approach and means to enable the realisation of successful systems" (INCOSE, 2011). Its emphasis is on defining customer needs, robust generation and traceability of system's requirements and functions via multiple viewpoints, both within and across its boundary's interfaces by maintaining system-of-systems context. A system often fails due to incomplete specifications of functions and

functional requirements at interfaces (Campean & Henshall, 2012). There are two key reasons to this that are often found in industry practice. First, partial views and viewpoints are often considered with no proper traceability and linkage in between manual documents managed and generated by designers working in different places and disciplines while modelling a system resulting in inconsistent set of requirements, and functions not only at one level but across multiple decomposition levels (Tate, 1999). Secondly, structural-based system decomposition is mostly practiced in industry (Jarratt, 2004). This causes late and costly changes to the companies due to issues of decomposition and integration of independent systems working together as a whole system. Thus, this demands for a structured approach underpinning systems engineering principals that could promote system-of-systems thinking across different disciplines both top-down and bottom-up with an emphasis on integrating yet applying consistently multiple modelling viewpoints for a system design in a robust manner.

1.2 Motivation

1.2.1 Research field

This work primarily revolves around the research of the Bradford Engineering Quality Improvement Centre (BEQIC) which has gained 10 years of enormous industrial experience by collaborating with automotive companies. The mutual collaboration work has resulted in an inspiring integrated Failure Mode Avoidance (FMA) framework based on a series of functional and interface modelling tools, suitable for automotive industry practice. The framework, in Figure 1.1, primarily developed in the philosophical context of “reliability is failure mode avoidance” (as outlined by Clausing, 2004) with a view to prevent failure modes early in the design process (Campean et al., 2010). The framework has also been discussed within the contexts of systems engineering in (Campean et al., 2013).

The research work by Campean et al. (2011; 2013) and Henshall et al. (2015) shows that system’s function and interface analysis with top-down decomposition in the context of failure mode avoidance practice is not well integrated nor fully aligned with the systems engineering process. The consequence of this is that industry ends up with mostly problems occurring at

interfaces and late changes as evident from studies in existing literature (Webb 2002; Dong 2002; Ford 2004).

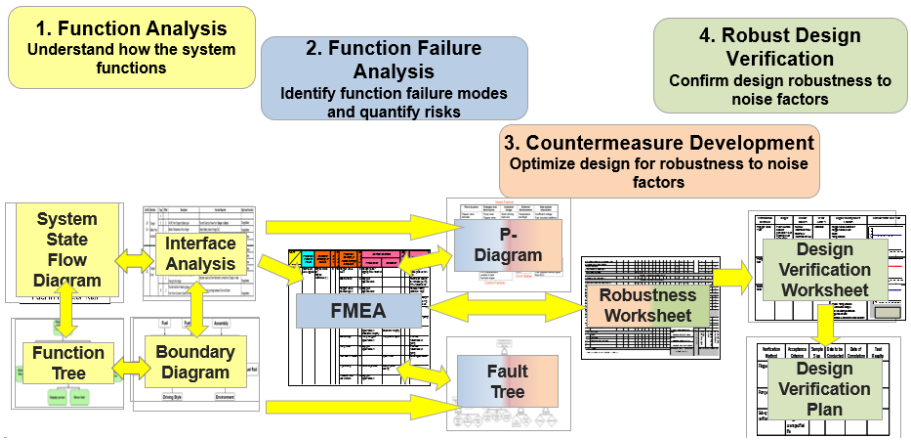
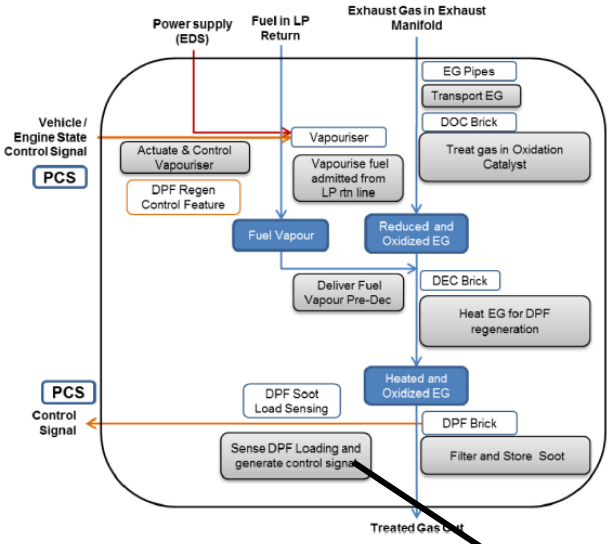


Figure 1.1 The BEQIC FMA framework (adapted from Campean et al., 2013)

Looking at aforementioned challenges in automotive industry, there was a fundamental need to strengthen the function and interface modelling tools to ensure that both solution independent and dependent analysis complement each other for top-down and bottom-up analysis. Campean et al (2011) resolved such challenges by introducing the structured function and interface-modelling tools as the basis for functional analysis in an integrated FMA framework for systems engineering design known as the 'system state flow diagram' (a graphical tool) and the 'interface analysis table' (a tabular template), as shown in Figure 1.1. The tools' schematics are also shown in Figure 1.2. The interface analysis table provides structured guidelines to identify functional requirements to manage the exchanges between two subsystems at solution-dependent stage and linking those to system's main flow related functions identified via states-based thinking in functional architecture at solution-independent stage via system state flow diagram.

Figure 1.2 illustrates interface analysis table for the 'exhaust after-treatment' system where interface functional requirements such as 'operate within environmental humidity range' and 'Be inert to corrosion' between subsystem and environment interface are linked with system's main functions 'sense DPF (diesel particulate filter) loading' identified via system state flow diagram along with its design implementation 'DPF soot loading sensing' subsystem. The

interface analysis table is concerned with the concrete implementations of the main functions that occur at subsystem boundaries.



Ref	Inter-face	Type	Effect	Metric	Description of Exchange	Function Required	Main Function
5-E8	SLS/ Ext Env.	E	-1	%/ml	Humidity/moisture/water	Operate within environmental humidity range	Sense DPF loading
		E	-1	N	Impact from debris i.e. rock	Withstand impact from road debris	Sense DPF loading
		P	-1	litres	Water	Be inert to corrosion	Sense DPF loading

Figure 1.2 System state flow diagram and interface analysis table for the exhaust after-treatment system (adapted from Campean et al, 2013)

These tools have been extensively taught within an automotive companies and also tested and applied by engineers within the same companies. Results reveal that the approach is applicable across multidisciplinary domains (Campean et al., 2013) and proved to be very useful at the physical system layers (i.e. from subsystems to component to manufacturing levels).

However, it is noted based on critical analysis that FMA framework is not fully integrated with the systems engineering design process activities/stages due to following three key reasons:

- ‘Requirements analysis’ stage in FMA framework is missing which comes before the ‘function analysis and allocation’ stage as per systems engineering standard, as shown in Figure 1.3. Firstly, this means that there is no supporting path or tool in FMA framework for analysing system’s services and operational interactions with its stakeholders at

system level (i.e. beyond its form) that helps in deriving the system requirements.

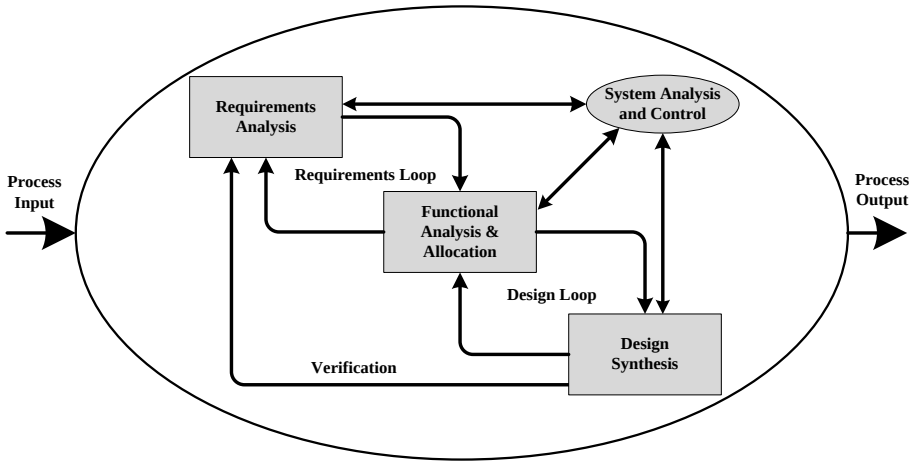


Figure 1.3 Overview of systems engineering process (DoD, 2001)

- Secondly, the FMA framework's 'functional analysis stage' (Figure 1.1) seems to have a set of tools that support systems engineering process later stages. For example, looking at system state flow diagram tool (Figure 1.1), it belongs to 'functional analysis stage' of systems engineering process (Figure 1.3) whilst the other tools in FMA framework's functional analysis stage such as system boundary diagram, interface matrix and interface analysis table for the purpose of design integration analysis belong to 'design synthesis stage' of systems engineering process. There is no formal path of allocating functions to subsystems which is an important activity in systems engineering process for creating and analysing system architecture.
- Thirdly, there is a lack of iterative loop paths for traceability between requirements, functional and physical analysis in FMA framework in contrast to systems engineering process. In short, there is no clear path for requirements loop that exist between requirements analysis and functional analysis & allocation as evident from Figure 1.3 and Figure 1.1. Also no clear path of design loop in FMA framework in Figure 1.1 that iteratively occurs as per systems engineering between functional analysis & allocation and design synthesis in Figure 1.3.

Hence, FMA framework is not fully discussed within the context of requirements analysis stage and other key iterative and recursive activities that are

recommended by systems engineering standards. Along with aforementioned challenges, some other key limitations are found in integrated FMA framework as follows:

1) The interface analysis table lacks a rigorous definition of the ‘interface’ and is limited in considering and modelling the multiple interaction viewpoints (or contents) at the interface.

2) The interface analysis table is mainly applied so far at solution dependent (physical architecture layer) where on one hand it supports strongly multiple exchanges/flows based analysis between two known subsystems of a system. It lacks to analyse system’s operational aspect (beyond form) driven by interactions analysis to achieve the goals related to its stakeholders on the other hand. In short, the interface analysis table supports viewpoints belonging to interaction modelling at physical/design stage but do not support viewpoints belonging to interaction modelling at requirements stage.

3) The various necessary modelling viewpoints (or concepts) available within engineering design and systems engineering for defining and analysing system interfaces and requirements have not been fully integrated with the interface analysis table and thus it has not been benchmarked with existing approaches available in literature.

4) A real world complex multidisciplinary system can possess either modular or integrated architectures or somewhere in between the two types (Ulrich, 1995; Uddin et al, 2016). The functional analysis stage of integrated FMA framework supports complex system’s modular architecture (one-to-one) development i.e. linear/ideal coupling between functional (solution independent) and structural (solution dependent) analysis at each successive decomposition level. It does not support or provide guidelines for the development of complex multidisciplinary system’s integrated architecture (i.e. one-to-many) in which one subsystem can implement multiple functions of a system and vice versa.

5) The interface analysis table and also the functional analysis stage of integrated FMA framework are not fully made compatible with other

systems engineering design tools such as use case diagram, context diagram, or sequence diagram for requirements capture, management, and in particular requirements trace-ability from ‘requirements analysis’ to ‘functional analysis and allocation’ to ‘physical architecture design’ stages that are recommended by systems engineering standards and approaches.

This analysis of the current practice of function and interface analysis and limitations of the FMA framework has provided the initial motivation for the research.

1.2.2 Thesis scope

The purpose of this section is to highlight the boundaries of the research this thesis intends to target:

- **External and internal Interfaces definition:** It is often needed to distinguish between internal and external interfaces of a system (Rahmani, 2012; Sage & Lynch, 1998). This becomes quite important when multiple levels of abstraction need to be defined for a system-of-interest. External interfaces occur between a conceptualised system-of-interest and the actors surrounding its environment such as user, other enabling systems and physical environment etc. (i.e. beyond system boundary) (Rahmani, 2012). Internal interfaces exist among the subsystems of a system-of-interest (i.e. with a system boundary) (Rahmani, 2012). This thesis focuses on definition of both types of interfaces for a system-of-interest with a view of deriving requirements.
- **System architecture and architecting:** System architecture has been perceived in different ways in literature. For example, architecture definition in (Ko, 2013) is restricted to decomposed components of a system belonging to physical design analysis only. This thesis focuses on architecture definition that involves system’s functions (functional view) and parts/subsystems with their interactions (physical view) to meet system requirements (requirements view) as found in (INCOSE, 2000; Bonnema, 2008; Tomiyama, 2012). Architecting is referred to a process of defining a system architecture (Bonnema, 2008).

- **Decomposition:** Divide and rule principle is generally employed and considered potentially useful to divide the whole problem into manageable small problems by finding respective sub-solutions in each sub-problem and thereafter integrating those sub-solutions into a system level solution (as discussed by Pahl et al, 2007; Tomiyama, 2012; Bonnema, 2008). However, the decomposition concept has got two key issues at hand (as outlined by Tomiyama, 2012): (1) lack of valid decomposition method to a variety of complex multidisciplinary systems and (2) the assumption of divide and rule principle with a view of sub-problems are independent to each other except for interactions at interfaces while in reality and in many cases sub-problems can have interactions for a large complex systems (Tomiyama, 2012). This thesis intends to develop an approach that could support multiple levels of system abstraction/decomposition.

1.3 Research Aim

The aim of this research is to develop an architecting approach centred on integrated interface modelling methodology to support system architecture and decomposition analysis.

The following objectives have been established in order to reach the research aim and looking at research motivation (Section 1.2.1) and the thesis scope (Section 1.2.2):

Objective 1: To carry out a systematic review of the academic literature on design process models in general, and a detailed review on the modelling approaches representing system interface and architecture modelling views and viewpoints;

Objective 2: To establish a concrete evaluation criteria, based on the information from the literature, aggregating the necessary mutually exclusive and collectively exhaustive viewpoints for defining system interfaces and architecture analysis; on this basis, a critical review of the existing approaches – focusing on scope and procedure, aiming to characterise their gaps and research hypothesis establishment in the contexts of the scope of this thesis;

Objective 3: To develop an integrated framework to support system architecture analysis centred on interface modelling approach across one level of abstraction (decomposition) via desktop case studies;

Objective 4: To validate the developed framework both theoretically and empirically across multiple levels of abstraction via real world case studies conducted with a set of independent engineers within an automotive company;

Objective 5: To review critically the experience of applying the developed framework from the theoretical and empirical analysis, to present the practical results and effectiveness of methodology thereby relating to research hypothesis, and to make recommendations for further work.

1.4 Thesis structure

This thesis is organised as follows:

Chapter 1 presents the research objectives based on the introductory literature review, research problems and current practice in automotive industry.

Chapter 2 reviews both prescriptive and descriptive design process models followed by modelling approaches for system design analysis with different views and viewpoints. The three critical conceptualised characteristics required for system architecture model are concluded in the end.

Chapter 3 develops a system architecture reference model based on information of Chapter 2. This chapter evaluates the scope and procedure of existing approaches based on visual criteria derived from the reference model in the context of the scope of this thesis. On the basis of identified gaps, the research hypothesis are established that set the ground for the developments of needed approaches in the fields of interface and architecture analysis for which research methodology is presented in the end.

Chapter 4 introduces a novel interface modelling methodology with a view that it can be applied consistently both on external and internal interfaces of a system. The methodology is validated with literature and desktop-based case studies at one level of decomposition.

Chapter 5 deals with integration of interface modelling methodology with architecture analysis activities thereby introducing a novel system architecture analysis approach which is also validated with literature and desktop-based case studies at one level of decomposition.

Chapter 6 validates the integrated architecture analysis framework centred on interface modelling with a real world engineering case study across its multiple levels of decompositions with a set of independent engineers of an automotive company. The gained practical results on the research developments are also related to research hypothesis in this chapter.

Chapter 7 accumulates and discusses the research findings from Chapters 2, 3, and 6. The developed approaches in this research are discussed in terms of their capabilities beyond the research scope in contrast to existing academic and industrial approaches (such as FMA framework).

Chapter 8 reviews key contributions and presents conclusions of this research, and also outlines a number of recommendations for future work.

2. Literature Review

2.1 Introduction

This chapter looks at both prescriptive and descriptive design process models in systems engineering and engineering design. The chapter then discusses the system modelling in the context of its architecture analysis which is defined and represented via numerous views in the research community. Based on this, an initial comparison framework is established that sets the foundation to explore literature review. After that the chapter reviews the viewpoints that are considered necessary in each view and understand the relationships between different views for modelling the system architecture and interfaces. This reveals the fact that many interconnected viewpoints are considered essential within various views for system analysis which often overlap by existing approaches. This sets the foundations for further critical analysis in the subsequent chapter.

2.2 System

The word *system* has a very broad meaning. In general, a system is defined as a set of interconnected components working together toward some common objective or more stated objectives (Kossiakoff et al., 2011; INCOSE, 2011). A system can be a natural system or man-made system: technological system and the societal system (Dong, 2002; Bartolomei, 2007). This research project focuses on *technological system*s that are engineered by humans.

Another word usually comes with a system definition in engineering design is *complexity*. It reflects the fact that elements (i.e. functional subsystems or modules, and components) in a system are diverse, interoperable and have complex *interface* relationships with one another (Kossiakoff et al., 2011). In today's world, an engineered system (e.g. automobile) consists of diverse mechanical, electronic and software subsystems where integration is pivotal not only *between* subsystems (e.g. propulsion subsystem and cruise control subsystem) but also *within* a subsystem (such as power subsystem and transmission subsystem within a propulsion system). The other complex systems include aircraft, weather satellite, electrical power plant, medical imaging, and telephone network systems. An enormous effort and good

collaboration among mechanical, electrical, and software engineers is needed for the design and analysis of such complex engineered systems.

A subsystem under consideration within each discipline (mechanical or electrical) can be a system itself composed of many other components which is a part of system or surrounding systems (Nomaguchi et al., 2006). Therefore, there can be various hierarchical or abstraction levels such as system, sub-system, components, sub-components and parts depending upon the organisation infrastructure (Suh, 2001; Wasson, 2006; Kossiakoff et al., 2011). INCOSE (2011) describes this fact with a term *system-of-systems* by discussing an air transportation system as shown in Figure 2.1**.** "System of systems applies to a system-of-interest whose system elements (i.e. subsystems or components) are themselves systems; typically these entail large scale inter-disciplinary problems with multiple, heterogeneous, distributed systems" (INCOSE, 2011).

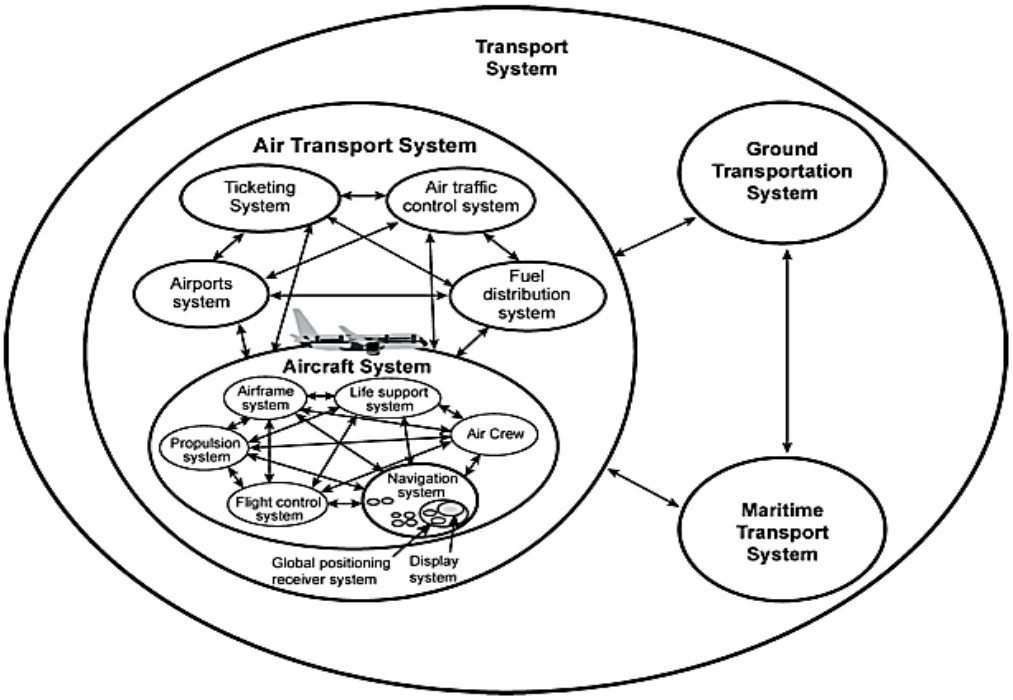


Figure 2.1 Possible systems of interest within and around the environment of air transport system (INCOSE, 2011)

To set the scope of analysis, a system boundary (a graphical diagram) is often used by system engineers to differentiate between elements that can be inside or outside the system under consideration and also to identify the *interaction*

relationships among those elements (JLR, 2014; Ford, 2004; Crawley, 2014). The interaction relationships between the interfaces of elements can be flows or *operands* related i.e. energy, material, and information (Pahl et al., 2007; Hubka & Eder, 1996; Otto & Wood, 2001) and/or *form* related, i.e. physical or spatial (Pimmler & Eppinger, 1994; Campean et al., 2011). The interactions may represent other business relations and dependencies (Muller, 2004), depending on the system's business and other lifecycle contexts. System engineers are dealing with such complexity not only at accomplishing stated objectives but also in determining their design hierarchy, both inter (i.e. between) and intra (i.e. within) elements' relationships and also providing functions in the process of developing large complex systems. Pugh (1991) formulates such system complexity as follows:

$$C_f = \frac{K}{f} \times \sqrt[3]{N_p N_t N_i}$$

Where *K:* scaling factor; *f:* number of functions to be performed; N_p: number of parts; N_t: number of different types of parts; N_i: number of connections and interfaces.

This clearly shows that system complexity is related to the number of parts, and their different types, and the connections and interfaces. But complexity does not end with system technicality and hierarchy but also is associated with *system architecting process* that defines and analyse a complex system architecture (Bonnema, 2008) i.e. a set of modelling *activities* and *views* of defining, generating, documenting, and maintaining the data that enable the implementation and production of *system architecture* (IEEE 1471, 2000).

2.3 System architecting process

2.3.1 Review of engineering design process

The existing state of the art for design process in existing literature is discussed and compared by many researchers (French, 1985; Hubka & Eder, 1996; Tate, 1999; Otto & Wood, 2001; Pahl et al., 2007; Ullman, 2010).

2.3.1.1 Purpose of designing

The purpose of the designing is to think ahead and identify issues in a distinct manner and thereafter addressing those issues in a clear statement in the problem space. The addressed statement is then utilised and explored in the solution space to develop system concepts. Hubka & Eder (1996) expressed the design process as “the transformation of information from the condition of needs, demands, requirements and constraints (including the demanded functions) into the description of a structure which is capable of fulfilling these demands” illustrated graphically in Figure 2.2. A *description* reveals and represents the architecture of a system. This is often referred as *architectural description* (Wasson, 2006). An architectural description (or model) can represent many views of a system. A *view* in turn may possess many associated *viewpoints* (IEEE 1471, 2000). Note that in the design process definition of Hubka & Eder (1996), requirements and constraints views come before functions.

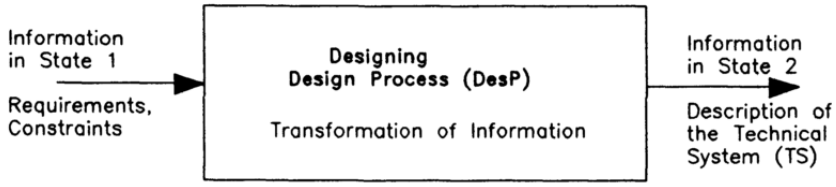


Figure 2.2 Black box diagram of the design process (Hubka & Eder, 1996)

2.3.1.2 Design process phases, stages, and activities

The design process catergorisation includes various design phases, stages and activities. *Phases* are usually related to a system’s lifecycle aspects such as designing, manufacturing, operation etc. as shown in Figure 2.3. Each phase is further broken down into stages and also each stage in turn into set of sequential activities. For example, in Hubka & Eder’s (1996) designing process, in Figure 2.3, the *stages* are planning, conceptualising, laying out and detailing whereas these stages are often referred as phases in other models e.g. by French (1985) and Pahl et al., (2007). These models in general define four to five stages of the design process or designing phase: (1) task clarification and specification development, (2) conceptual design, (3) embodiment design and (4) detail design.

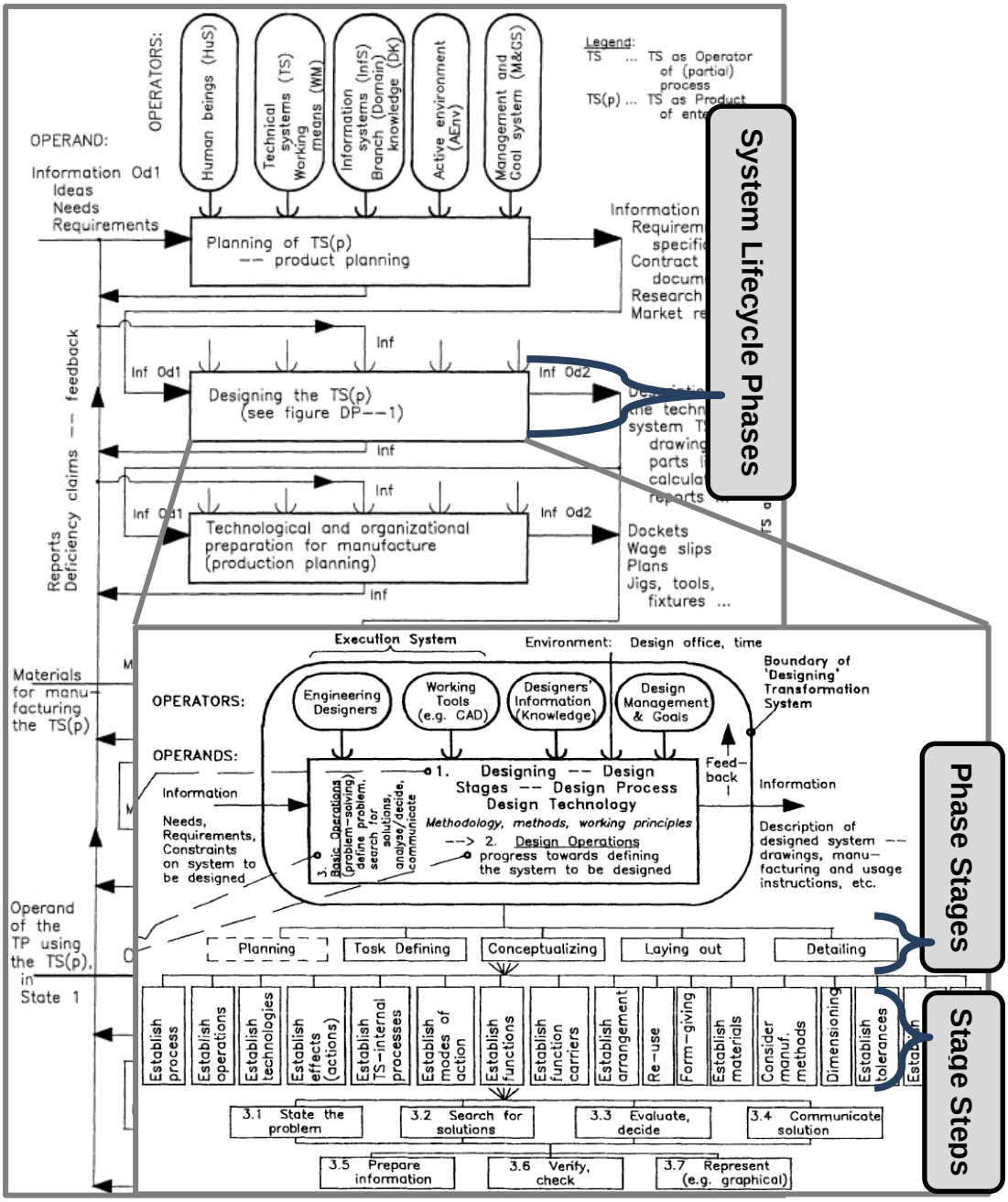


Figure 2.3 System design and development process phases, stages, and modelling activities (adapted from Hubka & Eder, 1996)

Each stage in turn involves several *modelling* steps and *activities* that are distributed among system developers from task clarification until working drawings and documents (see Figure 2.3). In the first stage, i.e. task clarification, an activity of establishing a system need is performed via market and/or quality experts which may involve analysing the issues in existing system or thinking ahead of features of a system never designed before. The description of the need can be vague for the designers so they resolute the need and study it further in order to design the required system. This activity is

often referred as clarifying the problem. The output of this phase is a statement of problem which is articulated in designer's language. A statement of problem can be a set of requirements list or specifications of a system to be designed that would indicate external environment and performance conditions.

In the next stage i.e. in conceptual design, designers explore the space of possible solutions in order to solve a given and well-defined problem. Also in this stage, the other activities are also carried out e.g. function structures are established, the best possible solution against technical and economic criteria is explored, and how this can be implemented and what resources for production planning are required.

Once an abstract concept is chosen, then it has to be developed further. French (1985) and Pahl et al (2007) call this Embodiment and Detailing phases. In these phases, an abstract concept is detailed that involves an actual physical implementation plan which may include a set of technical drawing sheets or analytical tools, a computer aided model, production plan, and facilitates layout.

2.3.1.3 Abstraction of knowledge

In phase-stage based sequential models, the design activities lead to an increase in information about the designed system on one decomposition level of hierarchy from an abstract to detailed knowledge step by step (Tate, 1999; Eisenbart, 2014). Different types of descriptions/languages or diverse documents are generally used to represent and describe a system such as concept sketches, structure layout, graphical diagrams, and specification sheets as summarised by Ullman (2010) in Figure 2.4.

	Level of abstraction		
Language	**Abstract**	⟶	**Concrete**
Semantic	Qualitative words (e.g., *long, fast, lightest*)	Reference to specific parameters or components	Reference to the values of the specific parameters or components
Graphical	Rough sketches	Scale drawings	Solid models with tolerances
Analytical	Qualitative relations (e.g., *left of*)	Back-of-the-envelope calculations	Detailed analysis
Physical	None	Models of the product	Final hardware

Figure 2.4 Incremental knowledge from abstract to concrete design (Ullman, 2010)

2.3.2 Review of systems engineering process

Systems engineering is an interdisciplinary research discipline and many models and standards have been established in it, as discussed and compared in detail by Estefan (2007).

2.3.2.1 Purpose of systems engineering

The purpose of systems engineering is “to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete design problem” (INCOSE, 2006).

2.3.2.2 Systems engineering process, phases, and activities

Unlike phase-stage based models, systems engineering related models and standards categorise a system lifecycle processes into technical (engineering), management, and organisational disciplines (Martin, 1997), in which design process is a part of whole systems engineering management plan.

The systems engineering process is applied iteratively at each system hierarchy level (Wasson, 2006). The engineering processes can be further laid down on highly recognised Vee-model into two phases: decomposition and integration as listed by Blanchard & Fabrycky (1998) in Figure 2.5. Unlike phase-stage driven design process models, systems engineering V-model shows realisation of system design at each engineering process definition by showing explicit relationship between decomposition (left side) and integration (right side) phases.

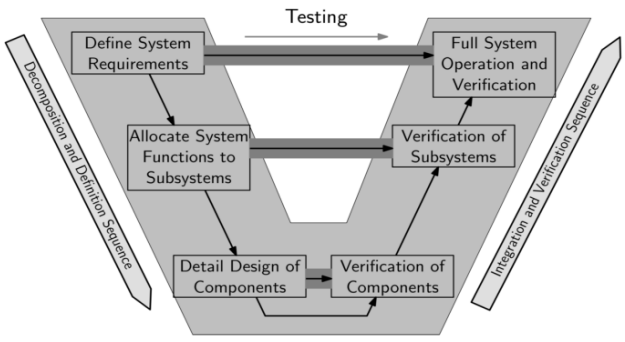


Figure 2.5 The Vee-model for systems engineering process (adopted from Bonnema, 2008)

On the left side, top-down vertically, the identified requirements on system level are cascaded down to subsystems and then from subsystem to components for detail design. Along with system requirements, at each decomposition level (left to right horizontally), test specifications are also specified and linked concurrently. These test specifications are used for qualifying the final system design at each level bottom-up.

Most systems engineering process standards in use today have evolved from the early days of DoD-MIL-STD 499, as shown in Figure 2.6, which is based on following set of processes: (1) requirements analysis (2) functional analysis and allocation (3) design synthesis (4) verification and (5) system analysis and control.

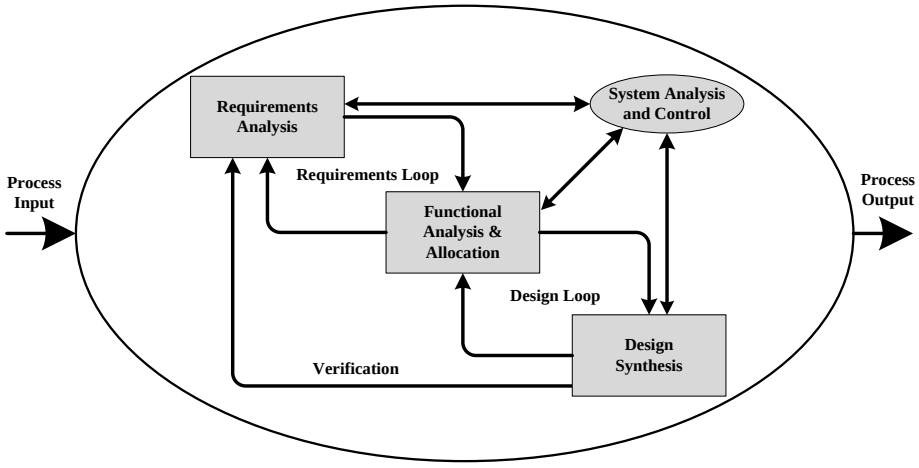


Figure 2.6 Overview of systems engineering process (DoD, 2001)

Like stages of phase driven design process models, each sub-process is composed of several modelling activities. Coherent with Figure 2.2, the *requirements modelling activities* associated with customer or stakeholder needs, and project mission requirements are used as an input to *requirements analysis process* (DoD, 2001; INCOSE, 2011). The customer or stakeholder requirements are transformed into a set of **functional** and **performance requirements** (i.e. technical requirements) as an output that define what the system must do and how well it must perform (DoD, 2001). The *requirements analysis process* also covers the activities of identifying environment, operational needs and design **constraints** (Kossiakoff et al, 2011). The outputs from requirements analysis are used as inputs to *functional analysis & allocation process* where *functional modelling* activities result in decomposing the **high level functions** (identified through requirements) into lower-level

functions thereby providing *functional architecture* of a system. The **lower level functions** are then linked with **performance requirements** of **higher level** (DoD, 2001). The iterative process of revisiting requirements i.e. traceability from functional analysis & allocation to requirements analysis (and vice versa) is termed as *requirements loop* as shown in Figure 2.6. The outputs from functional analysis & allocation process are used as inputs to *design synthesis process*. In this process, **hardware, software, and control elements** are chosen that could meet at least one functional requirement, and may support many functions (DoD, 2001) leading to a *physical architecture* of a system. In this process, other activities include evaluation of a number of alternative system solutions to achieve desired functions, interfaces among structural sub-elements against risk, and cost criteria (Kossiakoff et al, 2011). A mapping between functional and physical architectures is achieved via *design loop* which ensures that the developed **physical architecture** is capable of delivering **functional and performance requirements** (DoD, 2001; Kossiakoff et al, 2011). In verification process, the identified requirements at each level of development are verified. This is done by comparing the requirements against the chosen architecture and is referred as *verification loop* (DoD, 2001). In the end, through the iterations of such processes, an output of whole system engineering process is a complete description of ***system architecture*.**

2.3.3 System hierarchical levels and decomposition views

In systems engineering, a system-of-interest is often conceptualised in two ways: ‘black-box’ and ‘white-box’ views (Alexander & Zink, 2002). These two terms are generally used by test engineers in software testing. Black-box testing requires supply of valid set of inputs by tester to test the system’s functionality or use case and to examine the resulting expected outputs without looking into its internals or implementation details (Alexander & Zink 2002; Acharya & Vidhi 2012; Rational Software 2003). In contrast, a white-box test requires the tester to look ‘inside the box’ (Alexander & Zink, 2002; Rational Software, 2003) i.e. implementation details. The black-box term is very common in both systems engineering (see e.g. Kossiakoff et al, 2011; Rational Software, 2003) and engineering design communities (see e.g. Otto and Wood, 2001) in contrast to white-box term.

System hierarchy representation is commonly used in both engineering design and systems engineering communities (see e.g. Wasson, 2006; Dong, 2002) as illustrated in Figure 2.7a. It is the box views classification that is only used in systems engineering in the context of systems-of-systems or systems hierarchy representation (INCOSE, 2011; Alexander & Zink, 2002).

There is a relationship between system *hierarchy levels* and system *black-white box views* when it comes to problem *decomposition activity*. In order to elaborate this, three hierarchical levels are presented in a traditional style in Figure 2.7a. At a system level-0, a 'system-of-interest' is usually treated as a black-box. It contains inside many elements often referred as subsystems that generally belong to subsystem level-1 where these subsystems interact with each other to deliver the expected results to the external actors as illustrated in Figure 2.7b. However, it is important to note that system-of-systems decomposition activity has occurred only twice from level-0 to level-2 and not beyond level 2.

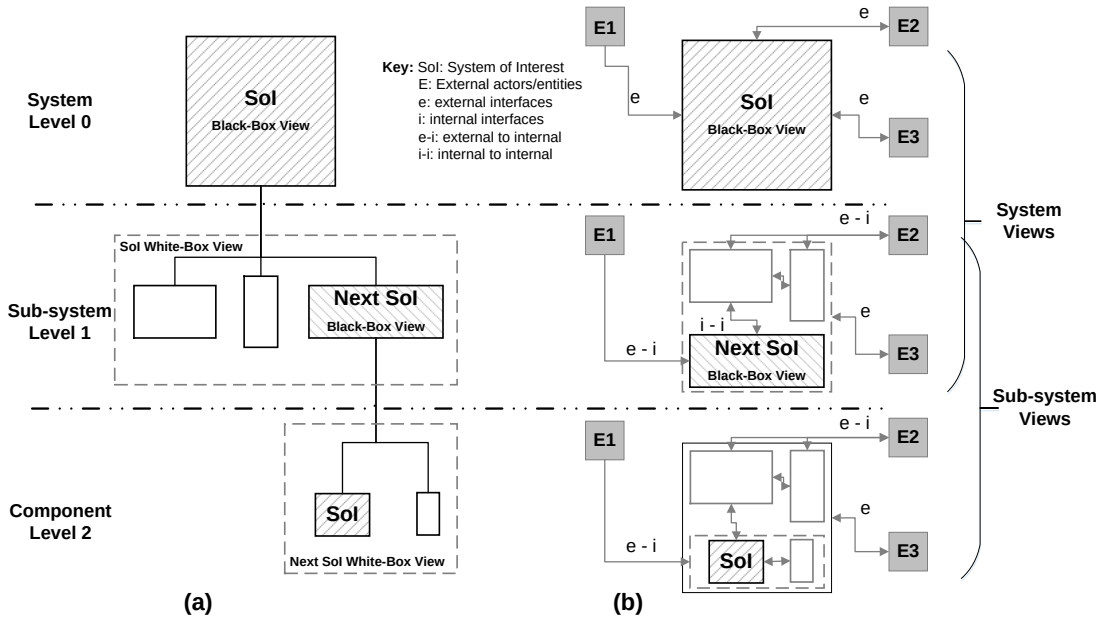


Figure 2.7 Relationship between a hierarchy level and conceptual views

It is also important to recognise that a system-of-interest can exist in any of these hierarchical levels and in there it can be approached from black-box and white-box perspectives. Each of the subsystem at sub-system level can be seen as a white-box for its (super) system-of-interest. Alternately, each of the subsystem at sub-system level in turn can become a next level system-of-

interest and thus can be perceived as black-box. For example, Figure 2.7b shows that a ‘next system-of-interest’ at subsystem level-1 is a white-box for a higher level-0 ‘system-of-interest’ while itself it becomes a black-box in the same level-1 and in turn if required may be decomposed to its white-box view that would reveal its subparts/components at component level-2. Crilly (2012) also discussed systems hierarchy (i.e. multiple levels of abstraction), as illustrated in Figure 2.7b, through a concept of *nested system* structure.

The black-box and white-box concepts can also be applicable to functional analysis in a way that a top-level function is often considered as a black-box with inputs-outputs (see e.g. Otto & Wood, 2001) while the decomposed sub-functions can be referred to white-box i.e. inside the black-box. Thus, a system (of interest) whether functional or structural related can be at any level within the system hierarchy where such views distinction is really useful from modelling and analysis perspectives. The relationship between levels and views is summarised in Figure 2.7b. The hierarchy Level-1 represents white-box view of a ‘system-of-interest’ and at same time it is responsible for black-box view of a ‘next system-of-interest’. Same logic applies to other hierarchical levels.

2.3.4 Summary of design processes models

Both systems engineering and engineering design processes share some following system design steps and also common modelling activities for the analysis of its architecture:

- customer needs transformation into measureable requirements;
- higher level functions decomposition into lower level functions;
- emphasize the progression of incremental knowledge from abstract to concrete detail.

However, there are some following key differences observed from allocation, and hierarchical levels perspectives:

- Iteration in design with clear traceability among the different views is explicitly emphasised in systems engineering models for architecture analysis. For example, traceability between requirements analysis and functional analysis at one decomposition level which is discussed vaguely in engineering design models.

- Systems engineering process acknowledge the use of similar set of architecture analysis activities across multiple hierarchical levels, i.e. recursive aspect whilst the engineering design process model do not explicitly as such.
- In systems engineering process model, an allocation of integrated viewpoints within a view of one hierarchical level to another elements in another view of lower level hierarchy is emphasized. This is also limitedly discussed in engineering design process models.

This thesis looks at both iteration and recursive aspects. Iteration occurs between multiple views of an architecture model whilst recursive aspect shows model applicability to any system-of-interest within a system, i.e. across multiple levels of abstraction (Bonnema, 2008; Wasson, 2006).

2.4 System architecture

2.4.1 System architecture definition and its modelling views

The definition of system architecture in literature varies and involves many perspectives. Research communities and researchers define system architecture as:

- "the arrangement of **functional elements**; the mapping from **functional elements** to **physical components**; and the specification of the **interfaces** among interacting **physical components**" (Ulrich, 1995).
- "the arrangement of **elements** and **subsystems** and the allocation of **functions** to them to meet **system requirements**" (INCOSE, 2000).
- "is an overall **system**'s **structure-behaviour** combination, which enables it to attain its **function** while embodying the architect's concepts" (Dori, 2002).
- **"**a graphical model or representation, such as an interpretative artistic rendering, a technical drawing, or a sketch, of a specific view of a system that communicates the **form, fit, or function** of a **system**, its operational **elements**, and **interfaces** as envisioned by its developer" (Wasson, 2006).
- "defines the **parts** constituting a **system** and allocates the system's **functions** and **performance** over its **parts**, its **user**, its **super system**

and the **environment** in order to meet **system requirements**" (Bonnema, 2008)**.**

- "describes concepts related to various **views corresponding to a domain** (discipline) specific view including **functional**, **behavioral**, and **structural** decomposition as the fundamental structure of the product" (system) (Tomiyama, 2012).

Table 2.1 summarizes five key *views* & their relationships from the perception of researchers, required for modelling and development of a system architecture. Therefore, modelling a system in general requires five key architectural views and traceability among the views on left side of Vee-model.

Table 2.1 Contrasting researchers' definitions for system architecture

	Ulrich (1995)	INCOSE (2000)	Dori (2002)	Wasson (2006)	Bonnema (2008)	Tomiyama (2012)
[R] Requirements		X			X	
[F] Functions	X	X	X	X	X	X
[B] Behaviours			X			X
[S] Structures	X	X	X	X	X	X
[I] Interfaces	X			X		

In the next sections, each view is explored and reviewed in the context of following questions along with relevant approaches in literature, as summarised in Table 2.2:

- View(s) – How each view is defined and perceived?
- Viewpoints – What other types and modelling viewpoints in each view are defined and considered?
- Views interconnectivity - How the information is traced between multiple views for architecture analysis?

Table 2.2 System modelling approaches and reviewed order in this chapter

Review order	Structure of current literature review
[R]	**System - Requirement view** • View definition & types • Approaches ✓ Quality function deployment ✓ Use case modelling • Summary of essential viewpoints
[F]	**System - Function view** • View definition & types • Approaches • Without operands & operators classification ✓ Sequential-functions-based analysis • With operands & operators classification ✓ Systematic design approach ✓ Theory of technical systems ✓ Functional basis approach ✓ State-flow-based approach • Summary of essential viewpoints
[R] → [F]	**Mapping of requirement and function views** • Approaches ✓ Constraints-functions matrix template ✓ Funkey coupling matrix ✓ Requirements-functions coupling matrix
[F] → [S]	**Mapping of function and structure views** • Approaches ✓ Axiomatic design's design matrix ✓ Icam / Integration DEFinition for function modelling ✓ Integrated function modelling
[F] → [B] → [S]	**Mapping of function & structure via system behaviour view** • Approaches ✓ Function-behaviour-state model ✓ Object-process-methodology model • Summary of essential viewpoints
[S]	**System - Structure View** • Approaches ✓ Context Diagram ✓ System boundary diagram / internal boundary diagram ✓ Design structure matrix
[I]	**System - Interface View** • Definition & types • Approaches • Diverse theories-based ✓ Contact & channel model ✓ Theory of affordance ✓ Port-based ontology • Interface documentation-based ✓ Use-case & events based interactions ✓ Flows based interactions ✓ Other tabular templates based approaches • Summary of essential viewpoints
[R]-[F]-[B]-[S]-[I]	**Descriptive reasoning models** • Four concept-based model • Five concept-based model

Based on definition, types, and the relevant approaches in each view, critical and essential viewpoints are drawn that are necessary for system modelling or architecture.

2.4.2 Requirement view on a system

Understanding of the system design problem requires translation of customer needs and objectives in the context of planned customer uses & environments into technical requirements for system functions (Ullman, 2010; DoD, 2001).

2.4.2.1 Customer needs

Customer needs are represented in the “language of the customer” (Ulrich & Eppinger 2003). The output from this is the structured list of requirements that may be vague and ambiguous (Burge, 2007). On another occasion, NASA (2007) categorises external drivers into two types: customer wants and interacting systems wants in the problem space for the complex system design. Interacting systems can be other stakeholders or enabling or interoperating systems (INCOSE, 2011). Thus in the customer need, it is not only end user but also other systems (or interested parties) that will interact with designed system.

2.4.2.2 Stakeholders related system attributes

Automotive industry defines 17 -18 attributes (note: numbers can vary) (often referred as vehicle level attributes) shown in Figure 2.8, that represent customer (end user and environment), corporate and regulatory requirements associated with vehicle system (Ford, 1997).

Engineering Atrributes List	
A1: Safety	A10: Vehicle Electronics
A2: Security	A11: Energy Management
A3: Accommodation & Usage	A12: Weight
A4: Thermal / Aerodynamics	A13: Cost
A5: Vehicle Dynamics	A14: Styling / Appearance
A6: Emissions	A15: Interior Climate Comfort
A7: Performance & Fuel Economy	A16: Product Compatbility
A8: Noise, Vibration, Harshness (NVH)	A17: Durability
A9: HMI & Audio Visual Performance	

Figure 2.8 System attributes list (adapted from Ford, 1997)

Stakeholder requirements grow tremendously over the period of time for larger complex systems such as automotive vehicles and often vary depending upon the stakeholder type and thus keeping the record of those in one deck can become a tedious task. For these reasons, automotive industry uses *divide & conquer* rule approach to successfully develop large systems. The different stakeholders’ requirements are grouped according to attribute types which is a

good practice from the perspective of dividing a problem into bits, conquering those and then integrating them as a final solution. All these requirements within different attributes can be seen as an interface requirements between vehicle system and its stakeholders (customer, corporate and regulatory). In essence, these attributes act as a bridge between stakeholders' needs viewpoint and system technical requirements viewpoint.

2.4.2.3 Technical requirements

The output of customer needs translation phase should be a structured list of technical requirements in a complete, consistent and correct manner which is often a challenging bit upfront in the design process as highlighted by researchers (Pahl et al, 2007; Ullman, 2010). According to Ullman (2010), "… all decisions are based on incomplete, inconsistent, and conflicting requirements/ information".

2.4.2.3.1 Definition of a technical requirement

A technical requirement, according to Kossiakoff et al. (2011), should specify "what the system must do, how well it must do it, and what constraints it must fit and … correcting inadequacies and quantifying the requirements wherever possible". The characteristics of good requirement are discussed by Hood et al. (2008). According to Hood et al. (2008), a requirement should be unambiguous, understandable, free of duplication, traceable, verifiable and correctly derived. A brief comparison between two viewpoints is shown in Table 2.3.

Table 2.3 Need and requirement characteristics (adapted from Burge, 2007)

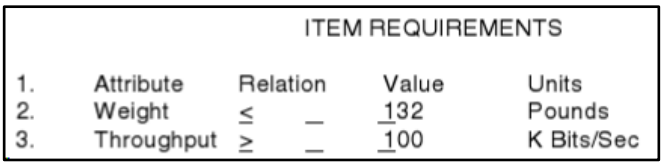
Customer Need	**Technical Requirement**
Requirement Characteristics	
General	Context Specific
Ambiguous	Precise
Un-measurable	Measureable / Verifiable

2.4.2.3.2 Grammar of a technical requirement

Ontologies based methodologies have been developed for writing correct, complete, and consistent requirements (Castaneda et al., 2010; Bijan et al., 2013), however the basic contents and structure for defining technical requirements remains same and is well defined by Grady (2006) as a *concept*

requirements list template, shown in Figure 2.9. The structure describes (i) the controlled item (or attribute), (ii) its value (with possible tolerance) and corresponding units of measure, and (iii) also the relationship between the attribute and value (such as equals or less than).

The system requirement sentence may involve use of 'subject+shall+verb+noun (item/object)' or 'verb+noun (object)' format in association with requirements content e.g. the system *(subject)* shall sustain *(verb)* object *(noun = attribute)* at a minimum *(relation)* rate of *(value) (units).*



ITEM REQUIREMENTS

	Attribute	Relation	Value	Units
1.	Attribute	Relation	Value	Units
2.	Weight	≤	132	Pounds
3.	Throughput	≥	100	K Bits/Sec

Figure 2.9 Concept requirements list example (Grady, 2006)

2.4.2.4 Types of technical requirements

A complete and consistent holistic categorisation model can be used for classifying and describing any set of system requirements (Burge, 2007) for following purposes: (1) to avoid documenting same requirement multiple times in the same document (Buede, 2009), (2) generation of overlapping requirements (Salado & Nilchiani, 2014) and (3) searching of system's specific requirements from a deck of large documents (Buede, 2009; Salado & Nilchiani, 2014). This thesis is primarily concerned with the functional and (non-functional) performance requirements. The definition of functional and performance requirements are given in Table 2.4. System level requirements have been defined and classified in many ways by various researchers (see e.g. Salado & Nilchiani, 2014; Buede, 2009; Burge, 2007) which are also clustered in Table 2.4, due to similar notation and definition.

Table 2.4 Definitions of types of a technical requirement

References	Types	Definition
Wasson, 2006	Operational Requirement	*"Operational requirements consist of high-level requirements related to system mission objectives and **behavioural interactions** and **responses** within a prescribed OPERATING ENVIRONMENT and conditions".*
Burge, 2007		*"Operational Requirements define the major purpose of a system (i.e. what it fundamentally does; its capability) together with the **key overarching constraints".***

Salado & Nilchiani, 2014	Functional Requirement	*"Requirements that define what the system must do in essence, or, in other words, what it **accepts** and what it **delivers** (i.e. expected transformation)".*
Burge, 2007		*"[…] specify what the system has **to do** in order to achieve the Operational Requirements".*
Malan & Bredemeyer, 2001		*"Functional requirements capture the **intended behaviour** of the system. This behaviour may be expressed as services".*
Buede, 2009	Input / Output	*"**Input requirements** state what inputs the system must **receive** and any **performance or constraint aspects** of each. **Output requirements** state what outputs the system must **produce** and any **performance aspects**"*
Wasson, 2006	Capability Requirement	*[Capability] "requirements specify and bound a solution space with functional/logical and performance actions each system entity or item must be capable of producing such as outcome(s), products, by-products, or services". Traditionally these requirements were often referred to as functional requirements.*
NASA, 2007		*"Functional requirements define what functions need to be done to accomplish the objectives".*
Salado & Nilchiani, 2014	[Non-Functional] Performance Requirement	*"Requirements that define how well the system must operate, which includes performance related to functions the system performs or characteristics of the system on their own, such as illities".*
Burge, 2007		*"[Non-Functional] Performance Requirements are associated with corresponding Functional Requirements and define how well a particular function has to perform – they are the constraints on that function".*
Wasson, 2006		*"[Non-functional] requirements relate to physical system attributes and characteristics of a system or entity".*
Wasson, 2006	Interaction / Interface Requirement	*"[Interface] requirements consist of those statements that specify and bound a system's direct or indirect connectivity or logical relationships with external system entities beyond its own physical boundary".*
Salado & Nilchiani, 2014		*"[Interaction] Requirements that define where the system must operate, which includes any type of operation during its life-cycle".*
Buede, 2009		*"[External interface] requirements deal with limitations placed upon the receipt of inputs and transmission of outputs by the interfaces of the external systems"*
Salado & Nilchiani, 2014	Resource / Requirement	*"Requirements that define what the system can use to transform what it accepts in what it delivers".*
Burge, 2007		*[Non-Functional Implementation] "Requirements define how a system is to be built in terms of specific technology. These may be specific requirements from the customer about a solution that they require or they may be legislative requirements".*
DoD, 2001 Bonnemma, 2008, Wasson, 2008	Allocated Requirement / Budget / Design To Requirement	*"A requirement that is established by dividing or otherwise allocating a high-level requirement into multiple lower-level requirements. Example: A 100-pound item that consists of two subsystems might result in weight requirements of 70 pounds and 30 pounds for the two lower-level items".*

2.4.3 Approaches for analysing system requirements

The development of correct and robust system level requirements from stakeholders is one of the primary challenges upfront in the system design. Quality Function Deployment, and Use Case Modelling approaches are prominent in this aspect (Bijan et al., 2013).

2.4.3.1 Quality function deployment

Quality function deployment is a systematic matrix-based approach for transforming customer ‘needs’ (What) into system ‘performance requirements’ (How). It involves four interlinked matrices known as quality tables in which information flows from customer needs till production requirements (Hauser & Clausing, 2009; Tapke et al., 2014), shown in Figure 2.10. According to Hauser and Clausing (2009) the first house maps the customer needs (end-user and other stakeholders related) with system engineering design requirements.

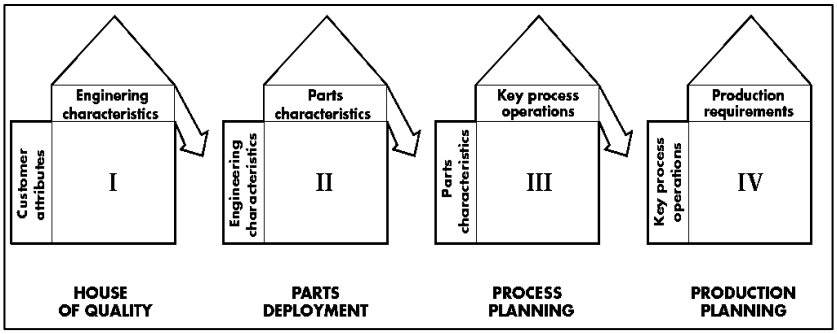


Figure 2.10 Overview of quality function deployment (Hauser & Clausing, 2009)

Electro-mechanical engineers have been using this approach successfully for years (e.g. Rahim & Baksh, 2003; Zheng & Pulli, 2007; Hassan et al., 2010). There are many advantages of it from the view of requirements analysis. For example, the first house maps customer needs to relevant engineering (performance) requirements along with rank of relationship category (i.e. small-1 to strong-9) for prioritizing requirements. The positive and negative interactions (or conflicts) between requirements are also marked in the top roof. The competitors benchmarking is also done. The key disadvantage of QFD is that it does not show system functions view and their distribution over performance requirements which is also essential (as discussed in section 2.3.2.2).

2.4.3.2 Use case modelling

Software engineers have introduced use cases for modelling a system (Cockburn, 2000; Bijan et al., 2013; Eisenbart, 2014). According to Daniels & Bahill (2004), it is difficult to understand the context of the system by considering only traditional ‘shall’ format requirements and relations between

them. The requirements should be traced to use cases within a system model in order to ensure a complete set (Buede, 2009; Hoffmann, 2011).
Use case modelling captures "who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals" (i.e. black-box) (Malan & Bredemeyer, 2001) or “with the internals” (white-box) (Cockburn, 2000). This is done initially via a use case diagram as shown in Figure 2.11. Bubbles indicate use cases (i.e. goals) and relevant actors are mapped to each use case via stick line. There can be sub-use cases to a main use case which are represented via include (i.e. necessary to base use case) and exclude (possible extension conditions to base use case) relationships .

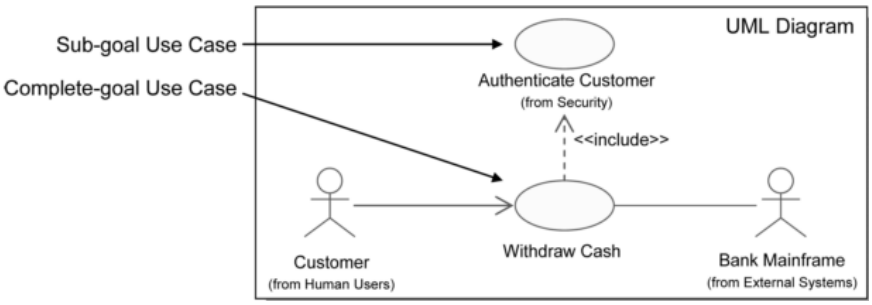


Figure 2.11 Simple use case diagram (Eriksson et al, 2008)

The next step describes a *scenario* that represent system behaviours with its stakeholders by considering their *interactions* under various conditions to accomplish the defined user goal(s) (i.e. *use cases*) (Cockburn, 2000; Daniels & Bahill, 2004). A *scenario* can represent a path of success or failure in which sequence of interaction events are described. Use case models are designed to serve as a bridge between stakeholders (i.e. voice of the customer) and the technical community (i.e. voice of the engineer) (Daniels & Bahill, 2004).

2.4.3.2.1 Daniels & Bahill’s use case modelling approach

Daniels & Bahill (2004) presented a hybrid requirements process based on use case modelling and traditional shall-statement requirements for complex, hierarchical systems. They advocated that system-level requirements both functional and non-functional can be derived within the specific use case thereby deriving relevant (sequence of) interactions as operations/events within various scenarios (i.e. main success, or alternate flow) as summarised in Figure 2.12. However, Daniels & Bahill (2004) discuss very little on the linkage between the two types of requirements.

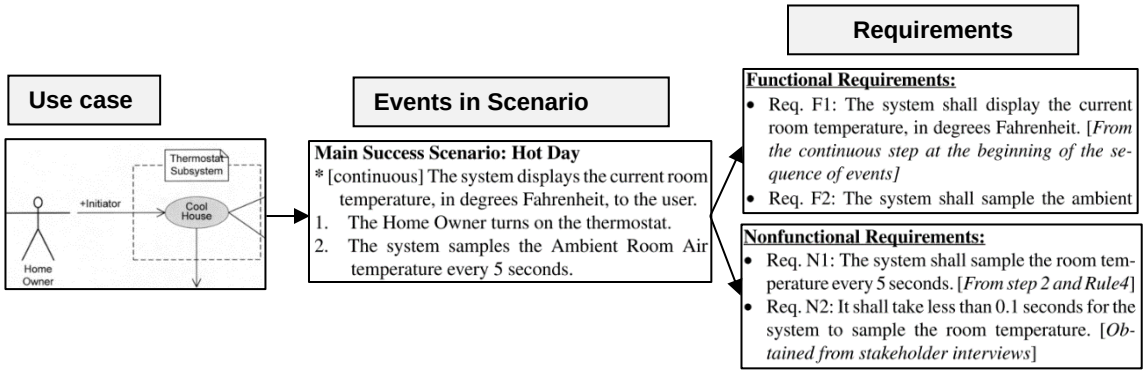


Figure 2.12 Deriving requirements via use case modelling (after Daniels & Bahill, 2004)

2.4.3.3 Summary

Two approaches can be summarised for requirements analysis. One approach is transform the needs directly into performance requirements where as in other approach goals are transformed into functional and non-functional performance requirements via scenarios thinking based on sequential operations/events. The key diverse viewpoints that are considered essential in these approaches for the analysis and derivation of system technical requirements are summarised in Table 2.5. The definitions of each viewpoint with relevant references are also provided.

Table 2.5 Viewpoints consideration within the requirement view-[R]

Reference	Viewpoint		Definition
Eisenbart et al, 2011	Customer Need		*"…refers to the identification or derivation of the fundamental need of the prospective users or the general market area to be addressed in relation to the new product or service to be developed".*
Ford,1997	System Attributes		*"These are vehicle level attributes that represent customer (end user and environment), corporate and regulatory requirements".*
Cockburn, 2000	Use Case (Goal)		*"A use case captures a contract between the stakeholders of a system about its behaviour".*
Wasson, 2006	Scenario		*"A hypothesized narrative that describes system entity interactions, assumptions, conditions, activities, and events that have a likelihood or probability of actually occurring under prescribed or worst-case conditions".*
Cockburn, 2000		Main Success	*"A scenario is a sequence of action and interactions that occurs under certain conditions, expressed without ifs or branching". "…The main success scenario is a case in which nothing goes wrong".*
Eriksson et al, 2008			*"Describes the "normal" way of achieving the goal stated in the use case name".*
		Alternative	*"Describes alternative ways of achieving the goal stated in the use case name".*
		Exceptional	*"Describe how different failures are detected and handled by the system".*

Malan & Bredemeyer, 2001	Interaction as (Sequence of) Operations / Event(s)	*"The interactions between the system and actors are structured into one or more steps which are expressed in natural language. A step has the form <sequence number><interaction>".*
Cockburn, 2000		*"An action step is the unit of writing in a scenario. Typically one sentence, usually describes behaviour of only one actor".* *"A message, a sequence of interactions, or a set of interaction sequences".*

2.4.4 Function view on a system

The function of a *function* word is to express system's functionality. The articulation of functionality of a system have been conceptualised in different ways by several researchers; still technical system functions are hard to express. There are following two key reasons for it;

- **Function definition***: Degree of abstraction issue* on a *function description* itself as mentioned by Umeda et al (1990).
- **Device/system views distinction:** Function description *with device's external* and *internal views* as highlighted and elaborated by Crilly (2012).

2.4.4.1 Definition of a function

From the function definition perspective, the function is expressed in different ways from the relationships between input and output of operands (material, energy, and information) to relationships between mating surfaces of subsystems / components (Umeda et al., 1990). The function definitions are summarised in Table 2.6. Stone and Wood (2000) provide a taxonomy of both operations (using 'verb') and flows (i.e. material, energy, and information using 'object/noun') as a function statement (i.e. verb-object format) which is well recognised and received in academia; however it has some limitations as discussed by Ahmed & Wallace (2003) from an industrial perspective.

Table 2.6 Definition of a function

Reference	Definition
Buede, 2009	*"A function […] is a process that takes inputs in and transforms these inputs into outputs. A function is a transformation, including the possible changing of state one or more times".*
Hubka and Eder, 1996	*"The function is a property of the technical system, and describes its ability to fulfil a purpose, namely to convert an input measure into a required output measure under precisely given conditions."*

Otto and Wood, 2001	*"A function […] is a statement of a clear, reproducible relationship between the available input and the desired output of a product, independent of any particular form… what it is to do; [it] is the simplest representation of the product, usually just a noun and an active verb."*
Dori, 2002	*"Function is what the system does and why it does it…. Function is a derivative of the system's goal – what service the system is expected to provide".*
Pahl et al, 2007	*"[…] it is useful to apply the term function to the intended input/output relationship of a system whose purpose is to perform a task.*
Ullman, 2010	*"Function is the logical flow of energy (including static forces), material, or information between objects or the change of state of an object caused by one or more of the flow". On another occasion…"Function happens primarily at interfaces".*
Soderborg et al., 2002	*"It describes the intended effect of a system's operation on the beneficiary user and the environment, not the operation itself. Thus multiple systems can fulfil the same function".*
US DoD, 2001	*"Functions are discrete actions (use action verbs) necessary to achieve the system's objectives. These functions may be stated explicitly, or they may be derived from stated requirements."*

2.4.4.2 Types of a function

From the system views perspective, Crilly (2012) stated, the function describes the system's *internal behaviour* within its boundary (i.e. at white-box) and also the system's *effects* on surrounding environment beyond its boundary (i.e. black-box); leading to two concepts: *device-centric functions* and *environment-centric functions* respectively, introduced by Chandrasekaran & Josephson (2000). The concepts are explicitly discussed (Vermaas, 2009; Erden et al., 2008; Brown & Blessing, 2005) and elaborated by various researchers with other types such as endogenous and exogenous functions in (Crilly, 2012), summarised in Table 2.7. Deng (2002) has classified function types into two depending upon abstract to concrete detail descriptions: *purpose function* (natural language) and *action function* (input-output operand related). The purpose function is articulated by designer depicting the stakeholder view (e.g. human/user) at abstract level whilst action function is also articulated by designer depicting the system view with concrete detail.

According to Chandrasekaran & Josephson (2000), "… a device is used because someone desired that something desirable happen outside the device. Thus, a central meaning of function is function as (desired) effect. However, functions are also often described in terms of the device's properties or behaviour, without any explicit mention of what the device might help achieve in the world outside it. Thus, functions can be described from a device-centric or an environment centric viewpoint, or even in a mixture of the two viewpoints." Chandrasekaran (2005) further explains that the device-centric functions are the means by which environment-centric functions are achieved i.e. the connection

between the two concepts. Erden et al. (2008) elaborate and show that the performance of *device centric functions* supports the performance of *environment-centric functions* which in turn fulfils *stakeholder's needs/services*. These two types of functions seem to be analogous to types given by Gzara et al. (2003) and by Sellgren & Anderson (2005) and Warell (2001), as summarised in Table 2.7.

Table 2.7 Types of a function

Reference	Types	Conceptualisation and perception of functions in relation to system boundary
Gzara et al (2003)	External functions	*"An external function expressing an action provided by the product to the environment. It describes what the product does to satisfy a user need"*
	Internal functions	*"An internal function describing the behaviour of product components in terms of how they contribute to realization of external functions".*
Brown & Blessing, 2005	Environment-centric functions	*"[…] in natural language one can describe the function of a device without knowing anything about its structure, or even about exactly what behaviours are at the D (device) to Ei (environment)* ***interface.*** *Those descriptions … closer to the user's desire. […] these descriptions as environment-centric (EC)".*
Crilly, 2012		*"…environment-centric functions need only refer to elements external to the device."*
Brown & Blessing, 2005	Device-centric functions	*"At the other extreme, the description can be solely in terms of the device: i.e. device-centric (DC)".*
Crilly, 2012		*"[…] that device-centric functions refer to the device's specific structural elements".*
Sellgren & Anderson, 2005 (Based on work by Warell, 2001)	Technical functions	*"Technical functions are associated with the flow, transformation, and storage of energy, materials, and information in the product. A technical function can be active, for example when it involves transporting or transforming something, or passive, for example when it involves supporting something".*
	Interactive functions	*"Interactive functions are associated with the interaction between the user and the product and communicate the usability and the attractiveness of a product".*

2.4.5 Approaches for analysing system functions

2.4.5.1 Without operators and operands/flows classification

The functions of a system are often modelled without considering explicitly its surrounding environment (i.e. without explicitly consideration of outside world). In following section, such modelling approaches are discussed.

2.4.5.1.1 Sequential-functions-based analysis

There are many approaches with different heuristics that recommend functional analysis of the technical systems based on arranging the functions in a time sequence (DoD, 2001; NASA, 2007; Kaufman & Woodhead, 2006).

Functional flow-block diagram: The functional flow-block diagram, in Figure 2.13, provides a function structure by arranging the network of functional

actions (or sub-functions) in the logical sequence that lead to the accomplishment of an overall function (Haskins et al., 1995). The functional blocks are numbered as per their hierarchical level. The two functions happening concurrently are represented via AND gate encapsulated in a circle whereas OR gate is used to show an alternative path for overall function execution. This function structure does not reveal a time duration to functions or between functions (NASA, 2007).

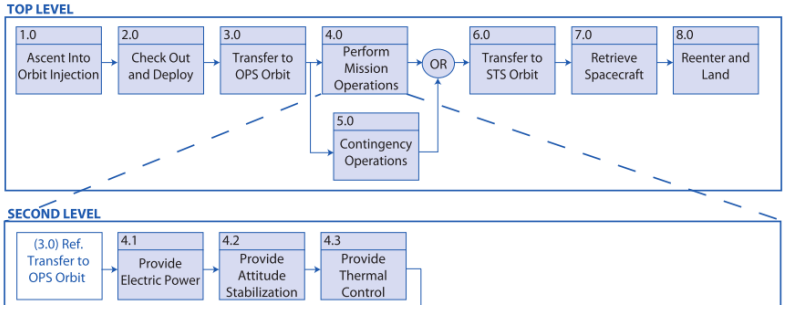


Figure 2.13 Excerpt of functional flow-block diagram of a spacecraft (adapted from NASA, 2007)

Function analysis system technique: The functional analysis system technique, in Figure 2.14, provides hierarchical decomposition of functions as well as arranges them in a logical sequence. The defined scope or problem of the system is analysed within two vertical dotted lines. This technique, however, requires tremendous effort in ordering the functions in relation to the operation of system (King & Sivaloganathan, 1998). Furthermore, it also provides no information on time duration to functions.

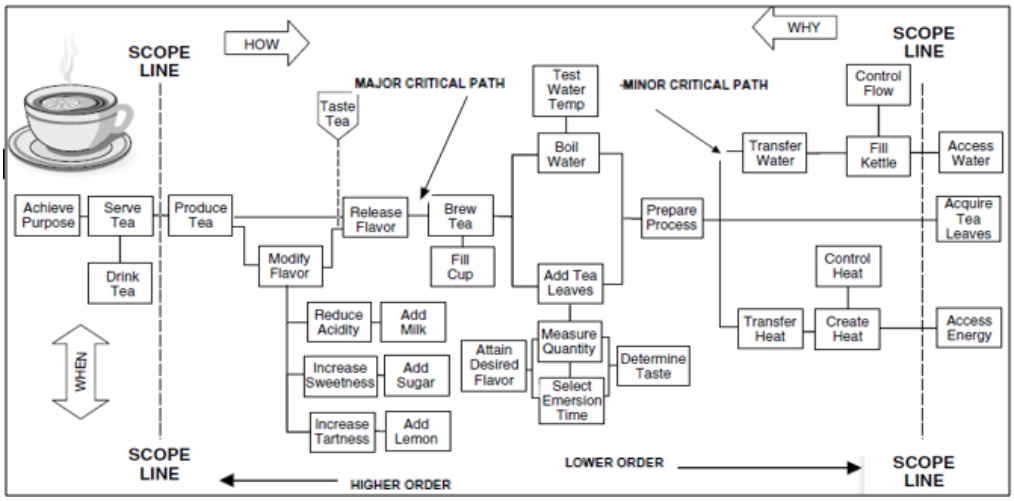


Figure 2.14 The functional analysis system technique model of brewing tea (Kaufman & Woodhead, 2006)

2.4.5.2 With operators and flows/operands

2.4.5.2.1 Systematic design approach

Pahl et al. (2007) define overall function (and the sub-functions) of a system in terms of input/output relations (see Table 2.6). According to systematic design approach (Pahl et al. 2007), the *technical artifact* that cannot operate on its own and in general is a part of larger system (i.e. *technical system*) where human, environment and other systems interact and influence it through *input (desired) effects* and/or *disturbing effects*, & in turn they get *feedback (desired) effects,* and/or *side effect* from artifact leading to further actions. Definitions of such different types of effect are summarized in Table 2.8.

It is discussed that these are the disturbing effects from environment (e.g. excess vibration and temperatures) that may cause side effects (e.g. deviation from desired performance and/or shape) from components within a system or from the overall system itself (Pahl et al, 2007). The side effects can have severe effect on human and other systems in surrounding environment that actually support or enable the *intended effect* of the technical system. The ***effects*** can be conceptualized into three fundamental flows i.e. *energy, material* and *information*. These are often referred as ***Operands*** by Hubka & Eder (1996).

Pahl et al (2007) recommends 'overall function' definition of a technical system thereby deriving it after analyzing the requirements list in Figure 2.15 (as discussed in section 2.3.2.4). In this regard, constraints are omitted and quantitative data is transformed into qualitative 'crux' statements and ultimately formulating a chosen problem in solution neutral way. The *overall function* from black-box is then decomposed into *sub-functions* (including *auxiliary functions* that support the main sub-functions) at white-box.

The emphasis is placed on ***solution independent function structure*** (i.e. functional architecture) establishment in association with operands relationships, as shown in Figure 2.15. However, in this function structure, when multiple flows are considered concurrently, it is not clear how are the lower level sub-functions generated (Zou & Du, 2013). Furthermore, the operands possess measureable properties (or attributes) which may vary from input to output

states (Hubka & Eder, 1996) and such information is critical to visualise from functions articulation perspectives which is also not seen in such function structure (Uddin et al., 2016).

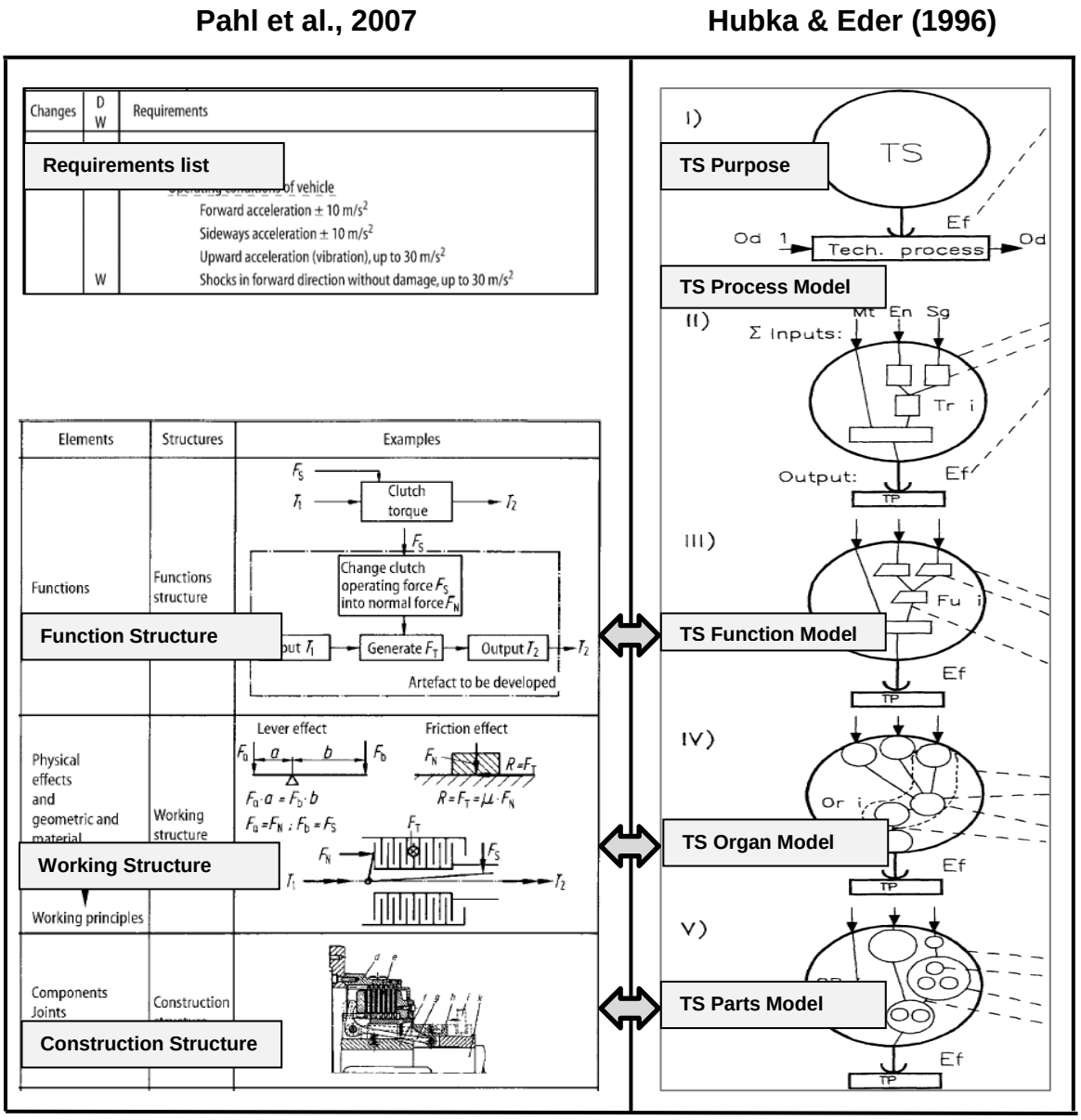


Figure 2.15 Flows related models (combination of Pahl et al. 2007 and Hubka & Eder 1996)

It is also unclear which lower level functions deliver which performance and functional requirements specified in requirements list when captured (at black-box around overall function or technical artifact) as shown in Figure 2.15. This is very essential step as addressed by systems engineering (DoD, 2001; INCOSE, 2011) (as discussed see section 2.3.2.2). One key fact becomes evident from this that operations (processes) layer in regard to stakeholders is missing

between requirements list and function structure in Pahl et al. approach. In the subsequent steps, the physical effects and working solution principles are then searched with sub-functions via morphological scheme in order to identify alternative working structures (i.e. physical architecture) that could deliver solution independent established function structure.

2.4.5.2.2 Theory of technical systems

Theory of technical systems, introduced by Hubka & Eder (1996), promote similar systematic design thinking like Pahl et al. approach but with the introduction of *process structure*, appearing before the function, organ, and component structures as shown in Figure 2.15. “These four structures, and an arbitrary number of further structures, can be represented for each technical system” (Hubka & Eder, 1996). However, the derivation of either a transformation process or function structure from requirements list is not explicitly discussed by Hubka & Eder in contrast to Pahl et al’s approach. Describing the relationships between four structures, it is also explicitly mentioned on two given examples by Hubka & Eder that the elements of both process and function structures are often in one-to-one correspondence i.e. “there is normally a coincidence between the TS-internal processes and the functions (capabilities) of the TS” (Hubka & Eder, 1996).

Few concepts and in particular three structures (i.e. function model, organ model, and parts model) by Hubka & Eder (1996) are analogous to Pahl et al. (2007) structures as illustrated in Figure 2.15.

2.4.5.2.3 Functional basis approach

The Stone & Wood (2000) key contribution is based on limitations in Pahl et al.’s (2007) functional descriptions for which they introduced operations and flows taxonomies. The function structure characterises main and supporting (auxiliary) sub-functions in terms of *flows* between them: material (M), energy (E) and information (I) and also in a logical sequence that help in achieving the overall defined intended function.

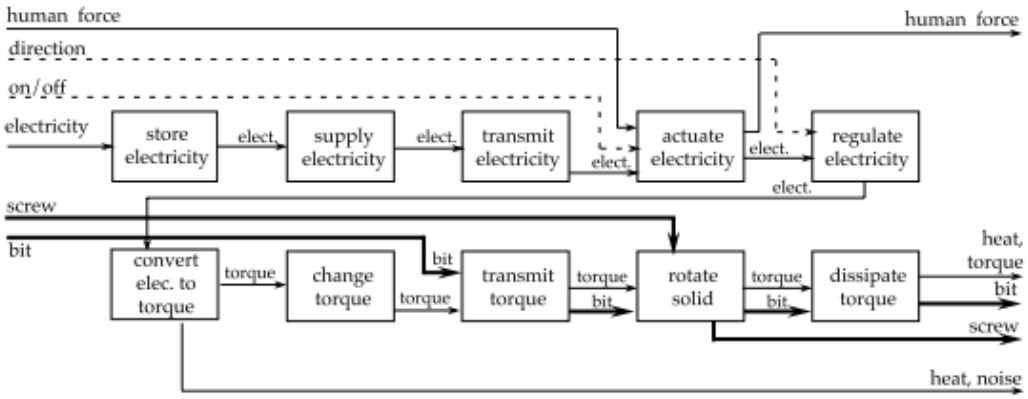


Figure 2.16 A functional model for power screwdriver (Stone & Wood, 2000)

Figure 2.16 shows power screwdriver function structure. However, the functional basis does not suggest guidelines for which functions are internal (device-centric) and external (environment-centric). This is also important for strong reasoning as discussed in Section 2.4.4.2.

2.4.5.2.4 State-flow-based approach

Many limitations are overcome in particular with multiple flows' states, and information representation, with another type of function structure, recently introduced by Campean et al., (2013) and Yildirim & Campean (2014), shown in Figure 2.17.

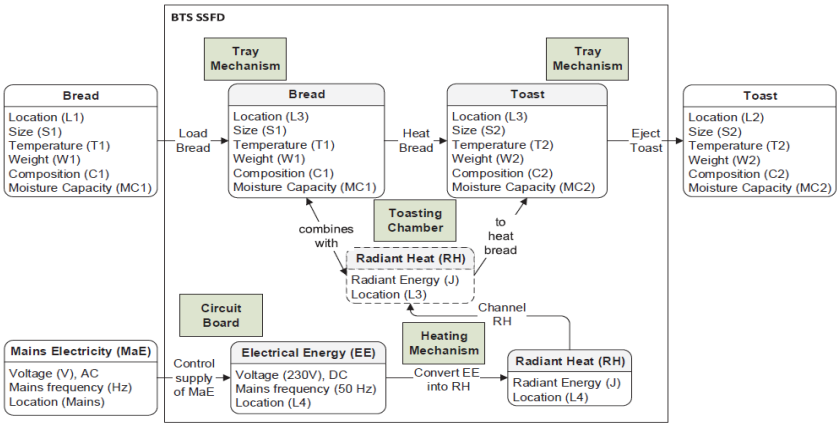


Figure 2.17 A functional model of toaster (Yildirim & Campean, 2014)

This type of representation identifies and arranges the functions associated with the transition of single main flow first and then secondary flows from their input states to output states by a set of measurable attributes. The system state flow diagram is a comprehensive approach for flows and functional modelling that

provides both visual clarity and detailed information (Uddin et al, 2016). Nevertheless it can be a time consuming effort due to strong thinking and sharp definition on operands' states and properties.

2.4.5.3 Summary

The key diverse viewpoints that are considered essential in the approaches for the derivation and distinction of system functions are summarised in Table 2.8. The definitions of each viewpoint with relevant references are also provided.

Table 2.8 Viewpoints consideration within the function view-[F]

Reference	Viewpoints		Definition
American Heritage Dictionary, 2016	Main Operand(s) / Flows		*"An **operand** is a quantity on which a mathematical or logical operation is performed".*
Hubka & Eder, 1996			*Operand […] object that is being changed in the transformation process (passive participant in the transformation) from an input state to a (preferably more desirable) output state.*
Solderborg et al, 2002			*"An operand is called a "transformee," emphasizing that a process transforms the object". […] the operand is what is affected or transformed (Soderborg et al, 2002).*
Eisenbart, 2014			*Operands are typically specifications of energy, material, and information.*
Hubka & Eder, 1996	Secondary Operands	Inputs	*"Secondary inputs […] (1) all necessary (desirable) additional inputs to the process, and (2) all undesired inputs (disturbances, contaminants, products of the environment, etc.".*
		Outputs	*Secondary outputs […] -- mostly undesirable outputs of the process, their nature and composition depend on the chosen technology"*
Harel, 1987	State		*"the condition of a given element which can be specified in terms of a set of value combinations".*
Dori, 2002			*"State is a situation or position at which the object can exist for a period of time".*
Hubka & Eder, 1996			*"State -- sum (vector) of the values of all properties of a system at a certain time. When observing a system, only the state of a selected group of properties is reported.*
Eisenbart, 2014			*"Representation of the states of actors or of operands before (input) and after (output) a transformation process"*
Hubka & Eder, 1996	Effect		*"Effects […] means of transformation – effects acting (actions exerted) on the operand, including supply of the necessary energy, auxiliary materials, regulation and control". & "In the design processes, the **effects** are to be considered as **goals** and the **structures** as means to achieve these goals.*
Eisenbart, 2014			*"Representation of the required physiochemical effects, which have to be provided, in order to enable or support transformation and/or interaction process(es)".*
		Intended Effect	*"Functionally desired effect in the sense of system operation".*
		Input Effect	*"Functional relationship due to human action on a technical system".*

Pahl et al, 2007		Feedback Effect	*"Functional relationship due to the action of a technical system on a human or another technical system".*
		Disturbing Effect	*"Functionally undesired influence from outside on a technical system or human that makes it difficult for a system to fulfil its function".*
		Unintended Effect	*"Functionally undesired and unintended effect of a technical system on a human or on the environment".*

2.4.6 Mapping of requirement & function views

The traceability between system requirements analysis and functional analysis is essential for system architecture as discussed in section 2.3.2.2. The key approaches are discussed in this regard.

2.4.6.1 Constraints-functions matrix template

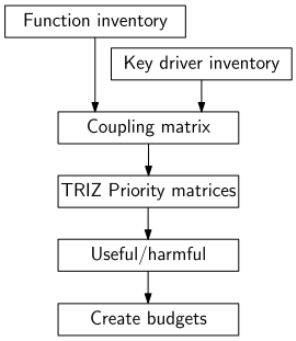
Tate (1999) and Friedman et al. (1998) proposed a matrix based approach to link the relevant functions (to them functional requirements from axiomatic design's language) to the five types of requirements (to them as constraints) for system architecture analysis in each hierarchical level, as shown in Figure 2.18. A change in constraint (requirement) can affect the whole design (Tate, 1999). Tate categorises following five types of constraint requirements: critical performance (seems as performance requirement), interface, global (as industry standards and safety regulations), project (as dictated by marketing and management and budgets) and feature (as constraints on the choice of design parameters). They argue that constraints defined at parent (top) level that are carried down and refined to lower hierarchical levels.

Constraint Table		Impacts:	FRs.		
Index	**Parent**	**Description**	**1**	**2**	**n**
Critical Performance Specifications					
C-1	Marketing	Meet throughput specifications	√	√	√
C-2	Marketing	Meet process specifications (arbitrary # of instances)	√		
Interface Constraints					
C-3	Marketing	Handle customer specified target objects	√	√	
Global Constraints					
C-4	Management	Maximise availability / reliability (minimise MTBF & MTTF)	√	√	
Project Constraints					
Feature Constraints					

Figure 2.18 Template for linking constraints-functions (adapted from Tate, 1999)

2.4.6.2 FunKey coupling matrix

Bonnema (2008) has introduced Funkey architecting approach that integrates functional view with requirements view for creating system architecture. Integration between two views is achieved via coupling matrix that connects functions to key drivers and/or performance requirements shown in Figure 2.19. Same matrix is used to allocate identified functions to both subsystems (i.e. internal actors) and surrounding actors (i.e. user and environment etc.) for system architecture creation**.** Each function and requirement can be further decomposed and relationships between them can be mapped using coupling matrix (Bonnema, 2008). Bonnema (2008) argues that coupling matrix is used for multiple purposes. First, it can be used to generate system budgets that help in deriving subsystem requirements. Secondly, the contradictions between system requirements that share same functions can be identified and solved using Theory of Inventive Problem Solving principles. The FunKey aforementioned steps are shown in Figure 2.19.



Function	cost/km	env.load	safety	convenience	Arch1	Arch2
drive vehicle						
– deliver force	×	×		×	U	PUT
– transmit force	×	×			PUT	PUT
maintain balance	×		×	×	U	PUT
maintain posture	×		×	×	PUT	U/PUT
create light						
– on road	×		×	×	PUT	Env
– on surroundings	×		×		Env	Env
– to other road users			×		PUT	PUT
steer	×		×		U	U/PUT
navigate	×			×	U	U
support user	×			×	PUT	U/PUT
provide energy	×	×			U	PUT

Figure 2.19 Funkey coupling matrix (adapted from Bonnema, 2008)

2.4.6.3 Requirements-functions coupling matrix

Buede (2009) provides a coupling matrix to support system architecture creation by integrating the functions and the input-output, external interface, and functional requirements of the system bounded by its external environment. The underpinning concepts of this matrix is based on architecting template given by Hatley & Pirbhai (1988). Figure 2.20 shows the coupling matrix that connects the functions with four types of requirements recommended by Buede (2009). Buede does not show the allocation of functions and requirements to

subsystems in the same matrix; rather opts ‘Integration DEFinition’ diagram (will be discussed).

Functions	Input/Output Requirements (A Sample)					
	Input Requirements		Output Requirements		Functional Requirement	External Interface Requirement
	The elevator system shall receive calls for up and down service from all floors of the building.	The elevator system shall receive passenger activated fire alarms in each elevator car.	The elevator system shall provide adequate illumination.	The elevator system shall open and close automatically upon arrival at each selected floor.	The elevator system shall control elevator cars efficiently.	The elevator system shall use a phone line from the building for emergency calls.
0 Provide Elevator Services	X	X	X	X	X	X
1 Accept Passenger Requests + Provide Feedback	X	X				X
1.1 Support Waiting Passengers	X					
1.2 Support Riding Passengers						
1.3 Support Passengers in Emergency		X				X
2 Control Elevator Cars						
3 Move Passengers between Floors			X	X		

Figure 2.20 Buede’s coupling matrix (adapted from Buede, 2009)

Buede (2009) recommends tracing of input requirements and output requirements to functions throughout the functional decomposition. Note that Buede’s interface requirement specifies a mean (an external solution/design element) that can be used for a specific job to deliver system functions.

2.4.6.4 Summary

This section has discussed that defined requirements of a system (at black-box) are allocated/mapped with its functions before looking into solutions (internal working structure at white-box). The next sections discuss how functions are allocated/mapped to structural (internal) solutions.

2.4.7 Mapping of function and structure views

One of the key steps in system architecture analysis is to allocate the functions to system’ subsystems. Following section discusses such approaches.

2.4.7.1 Structure *follows* function - modelling approaches

2.4.7.1.1 Axiomatic design matrix

The *function* domain is often considered as the intermediary between requirement and *structural* domains. In the Axiomatic Design approach, a product is modelled hierarchically via a zigzag procedure between functional requirements and design parameters of functional and physical (views) domains respectively (Suh, 1998). According to Suh (1998), design parameters may be *physical parts*, parameters or assemblies. The next level functional design

decomposition is based on the chosen implementation at one level above (Suh, 2001). However, it is also recognised by researchers that concrete decomposition operations on the system description have not been explained (Tate, 1999; Komoto & Tomiyama, 2011).

Axiomatic design provides design matrix that connects two viewpoints; the 'functional requirements' of a system to 'design parameters' and follows two axioms i.e. (i) maintain the independence of functional requirements (ii) minimise the information content. These axioms and design matrix provide a robust framework for evaluating design decisions at each level of system hierarchy, shown in Figure 2.21. It should be noted in Figure 2.21 that functions are decomposed based on chosen implementation at one level above.

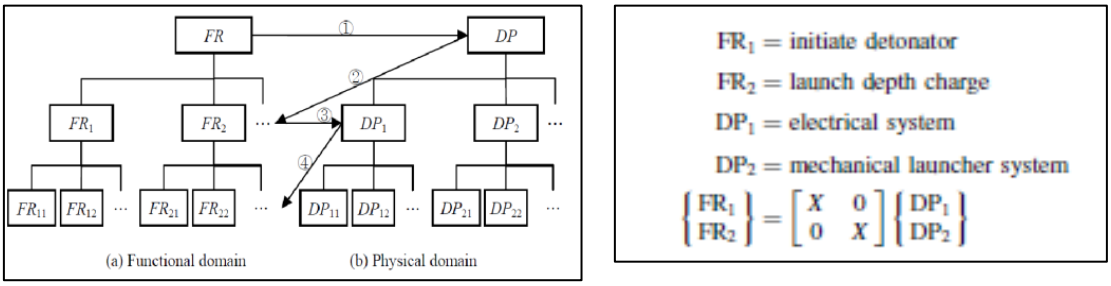


Figure 2.21 The functional & physical decomposition hierarchy and mapping via design matrix (adapted from El-Haik, 2005)

2.4.7.1.2 Icam / Integration DEFinition for function modelling

The Integration DEFinition diagram, in Figure 2.22, provides function structure of a technical system in which each decomposed transformative function (or activity or process in a box) includes a set of input items (entering left), output (leaving right) items, controls (on top) and mechanisms (from bottom) (Kossiakoff et al, 2011). The inputs represent receiving items from the user or the other system whilst the outputs represent the items that are sent to other systems by the system-of-interest. Input-output items can be physical objects or data or information related (Buede, 2009). Mechanism reveals the means (solutions or subsystems) to achieve a function whereas controls support or guide the transformative function.

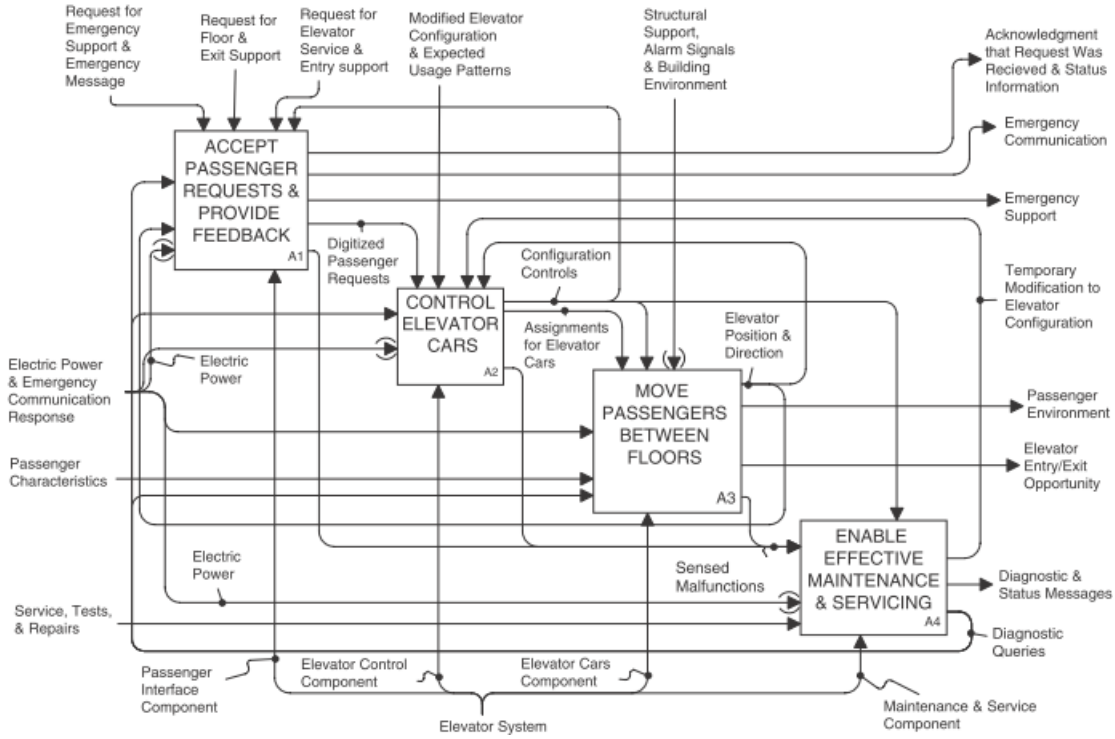


Figure 2.22 A child diagram of elevator for its top level function (Buede, 2009)

Integrated DEFinition diagram provides strong hierarchical structure and maintains consistent detailed information, due to additional controls and mechanism aspects. Nevertheless, it can be a time-consuming process and in particular the integration difficulty with other design related methodologies (Durugbo et al., 2011).

2.4.7.1.3 Integrated function modelling

Eisenbart (2014) proposed a comprehensive approach based on collective six viewpoints from literature and industrial practice that are considered useful across multidisciplinary engineers to support system development based on functional modelling as shown in Figure 2.23. The approach integrates the central *process* view with the other five viewpoints: *use cases*, *actors*, *states*, *operands*, and *effect* viewpoint via Domain Mapping Matrix & Design Structure Matrix, shown in Figure 2.23a. The design structure matrix is used for modelling the dependencies between a single domain e.g. for actor viewpoint whereas domain mapping matrix is used for mapping the dependencies across two different viewpoints e.g. use case and process flow. Eisenbart (2014) groups the definition of function and behaviour views of the technical system into a

process view and classifies it into two: *technical processes* and *human & interaction* processes (see Figure 2.24).

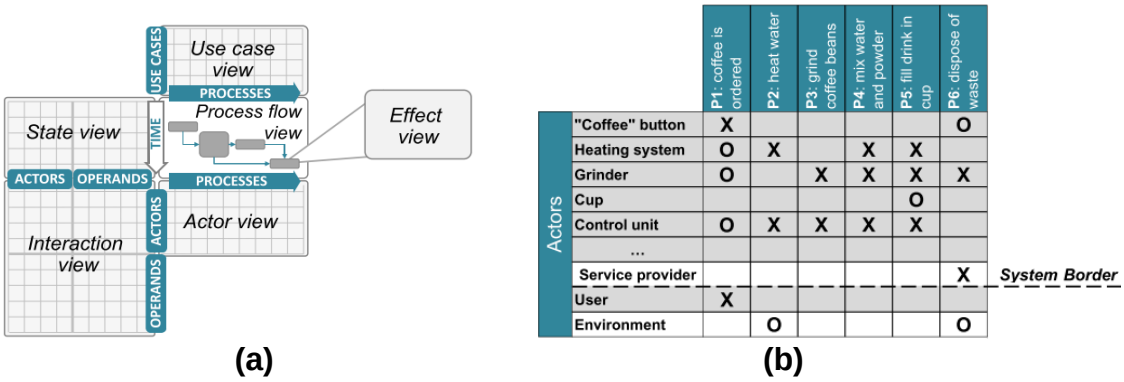


Figure 2.23 The integrated function modelling approach (adapted from Eisenbart, 2014)

Entities		Description
Use Case		Different cases of applying the system. This is typically associated to the interaction of actors with the system under development, which may require subsequent transformation processes to take place. The associated set of processes lead to an observable result, in order to provide some kind of value to users.
Process	Transformation process	Processes executed by actors, which (from the designers' perspective) are part of the system under consideration and may lead to a change of state of actors or of operands. *Technical processes* are transformation processes related to technical sub-systems; *human processes* are related to stakeholders (thus, including service activities).
	Interaction process	Representation of interaction processes of actors, which (from the designers' perspective) are *not* part of a system, with actors, which *are* part of the system under consideration.

Figure 2.24 Interaction and transformation processes definition (adapted from Eisenbart, 2014)

This means that function and behaviour views are implicitly covered underneath process view and rather appear explicitly with *effect viewpoint*. These processes are allocated to actors both internal (technical subsystems) and external (stakeholders and environment) via matrix, as shown in Figure 2.23b.

2.4.8 Mapping of function & structure via system behaviour view

Several researchers (Gero, 1990; Umeda et al., 1996) argue that function to structure mapping require interpretation of physical *behavioural view* for conceptualising the technical system development which is perceived in various ways by several researchers. This means that a research community believes that an indirect transition is the requirement between function and structure views which contradicts the approaches discussed earlier that recommends achieving either direct transition or perhaps indirectly with the consideration of

implicitly behavioural view or elements e.g. in integrated function modelling approach.

2.4.8.1 Behaviour *follows* structure *follows* function: modelling approaches

2.4.8.1.1 Function-behaviour-state model

Umeda et al (1996) introduced function-behaviour-state model to represent the relationships among function, behaviour and state (different notion/viewpoint for an entity) views by using electro-mechanical systems. They argue that function "is used in different degrees of abstraction; i.e. from relationships between input and output of material, energy, and information to relationships between surfaces of mechanical parts" (Umeda et al., 1990). They define function as "a description of behaviour abstracted by human through recognition of the behaviour in order to utilise it" (Umeda et al., 1990) and argue that a function cannot be represented independent of behaviour. The functions (e.g. to discharge electricity) are related to behaviours (e.g. electrical discharging) and it is the physical feature that embodies the function.

Physical feature in turn consists of entities nodes, their relation node, and a physical phenomenon, as shown in Figure 2.25a. A behaviour is defined as 'sequential state transitions along time'. In function-behaviour state model, behaviour is perceived as the 'physical phenomenon' that causes the change of the 'states' of entities of the system (Umeda et al, 1996), shown in Figure 2.25**.**

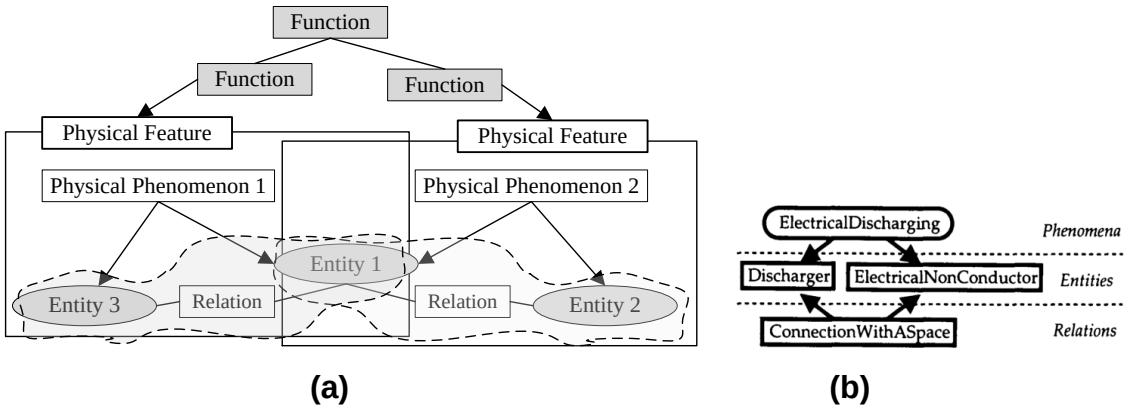


Figure 2.25 Function behavior state model (adapted from Umeda et al, 1996; Uddin et al., 2016)

For example, in Figure 2.25b, the sequential state transitions of an entity 1 (e.g. discharger) from its one state to another occurs while having certain relationship

with another entity 2 (e.g. electrical-non-conductor) with behaviour description (e.g. electrical-discharging). A physical phenomenon occurs on entities. The structural relationships between entities is represented via relationship node.

2.4.8.1.2 Object-process-methodology model

Dori (2002) has introduced a methodology that represent interactions in a system and applicable to electrical, informational, mechanical and human components. The methodology uses three key elements: objects, processes and states viewpoints to model the system and the relationships between them via several input/output and other structural links. This is graphically represented via Object-Process Diagram in Figure 2.26a, and its corresponding textual description set via Object-Process Language in Figure 2.26b.

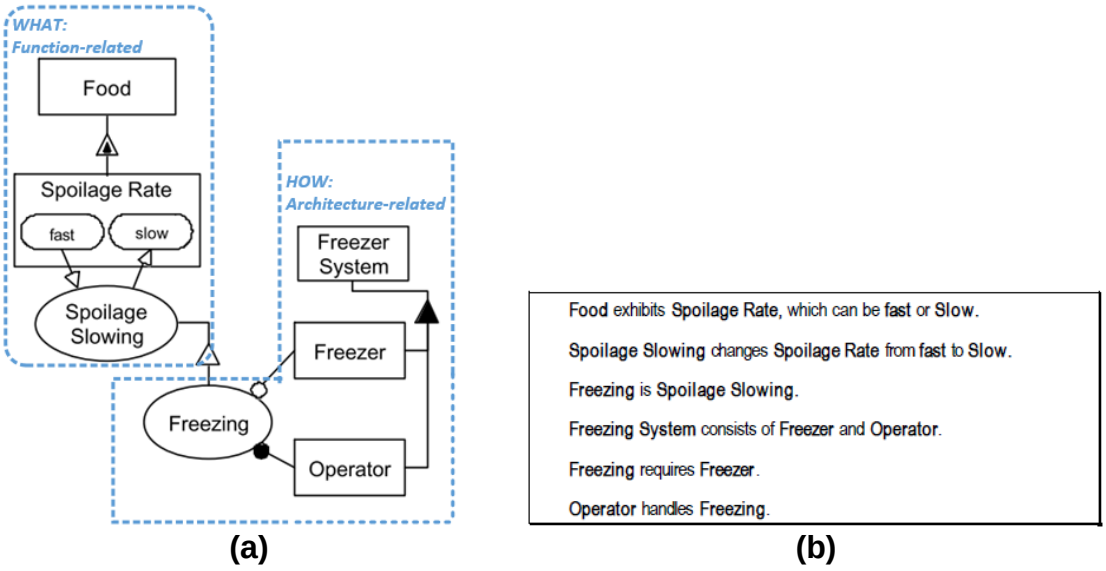


Figure 2.26 Object process methodology scripts (adapted from Soderborg et al., 2002)

According to Dori (2002), it is a methodology that “integrates function, structure, and behavior in one model”. In this methodology, in Figure 2.26, behaviours describe the processes (e.g. spoilage slowing) that cause the transition of ‘states’ of an operand (e.g. material related ‘food’) and also describe the process (e.g. freezing) performed by the entities (e.g. operator, and freezer) of the system as discussed by Soderborg et al. (2002), and shown in Figure 2.26. The two processes (i.e. related to an object i.e. operand flow and to system structure related) can be interlinked using generalisation link.

Dori (2002) refers system dynamics as behaviour and discuss clear difference between function and dynamics (behaviour) with many examples. Function is the derivative of system's goal i.e. what service it provides whereas the dynamics is how the system's structure behaves (changes over time) to perform its function (Dori, 2002). For example, in his example, both sundial and clock can achieve the function i.e. 'to tell the time of the day'. However, their achiectuire is different i.e. their structure-behaviour combination is different. Sundial is a static fixed device that is not easily portable whereas digital clock is portable thus both possessing different behavioural interactions (as effects) with a human (as surrounding actor).

2.4.8.2 Summary

From previous section, it is discussed different approaches perceive behaviour view in different ways, summarised in Table 2.9, in order to achieve the transition from function to form structure. Note that behaviour is also classified in many ways as summarised with definitions in Table 2.9. According to Gero (1990), the expected behaviour is used in the selection of structure to exhibit the intended behaviour to meet functions while actual behaviour is derived from chosen structure. Sellgren (2006) expands actual behaviour definition as given in Table 2.9

Table 2.9 Types and viewpoints consideration within the behaviour view-[B]

References	Types		Definition
Gero et al, 1990	Behaviour	Expected	*"the expected behaviour (Be) is used in the selection and combination of the structure…".*
		Actual	*"…is derived from the structure".*
Sellgren, 2006	Actual Behaviour	Accidental Behaviour	*"Actual behaviour can be decomposed into intended, unintended (side effect) and accidental behaviour (faults or errors)". "An accidental behaviour is an unintended behaviour that is caused by an accidental relation or interaction between product features (e.g. a cable placed too close to a hot motor block).*
		Intended Behaviour	*"Intended behaviour is considered as the means by which (or how) a function is achieved and it expresses physical state transitions or physical phenomena (principles)"*
		Unintended Behaviour	*"Unintended behaviour is a side effect that may require additional sub function to counteract, eliminate, or reduce the undesired side effect."*
References	**Viewpoint**		**Definition**
Umeda et al, 1996		Dynamic	*[Entity]…"sequential state transitions along time".*

Dori, 2002	Behaviour as change in state/structure		*"…dynamics (or the system behaviour) is how the system acts or operates to attain its function. […] how the system's structure behaves (changes over time) to perform its function"*
Rerverso Dictionary, 2016		Static	*"not active or moving; stationary"* *"(of a weight, force, or pressure) acting but causing no movement"*
Gedell et al, 2011	Behaviour as effects but actions / processes based		*"A system's behaviour is a result of the system and its interaction with its surroundings"*
Rosenman & Gero, 1998			*"the artefact's actions or processes in given circumstances of the natural environment".*
Eisenbart, 2014			*"Representation of the required physiochemical effects, which have to be provided, in order to enable or support transformation and/or interaction process (es)."*

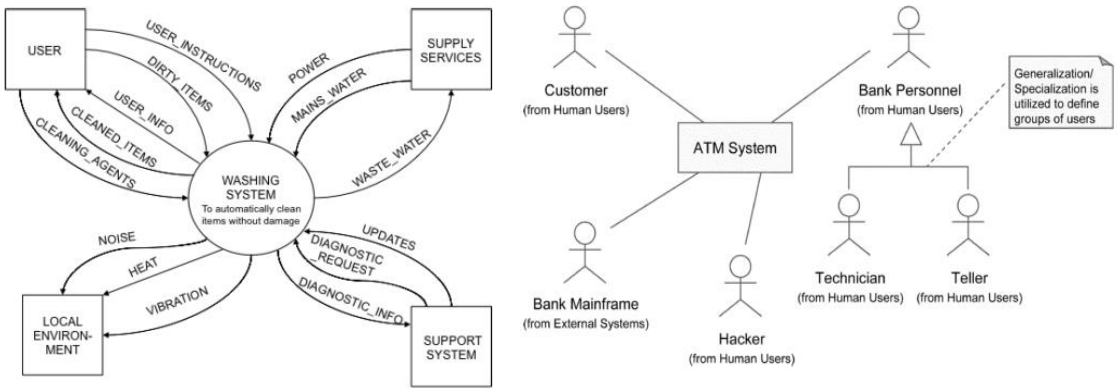
2.4.9 Structure view on a system

A system-of-interest may consists of internal elements such as subsystems (or modules or organs) as well as external actors such as user, environment and other systems that interact with it. This section looks at approaches that deal with system's internal and external structures.

2.4.9.1 Structural modelling approaches

2.4.9.1.1 Context diagram

A context diagram provides a high-level functional model (at black-box) that defines and visualises the boundary of the system-of-interest and its interactions with the critical external actors (such as users, other systems and environment) (Burge, 2011). It defines the system-of-interest (i.e. bubble in the centre) with no details of its interior functions or components (Kossiakoff et al, 2011). This is the pivotal difference in comparison to typical system boundary diagram (which is discussed in next section). The input-output interactions (i.e. unidirectional or bidirectional arrows) to/from system with its interacting external actors are represented in terms of flow quantities (i.e. material, information, and energy) as shown on left side in Figure 2.27a. Some researchers omit such input/output and services information in order to just visualise boundary and the surrounding actors around system-of-interest and then using other sets of tools to document the data (Eriksson et al. 2008). Each external entity or actor can be further categorised using generalisation relationship, as shown on right side in Figure 2.27b. It is discussed that identification of external actors is the hardest step that requires a group of team members.



(a) **(b)**

Figure 2.27 Context diagram for representing system external structure (adapted from Burge, 2011 & Eriksson et al., 2008)

2.4.9.1.2 System boundary diagram

This diagram is a graphical representation tool that describes the sub-systems or components within a system boundary (i.e. white-box or internal structure) and their interactions between them and across its neighbouring systems (Ford, 2004) as shown in Figure 2.28. The dashed line represents the system boundary and anything inside it is considered as area of interest (SGTS, 2012). The boxes represent sub-systems within a system and the arrows represent dependency and inputs-outputs of each subsystem with respect to another sub-system, as shown in Figure 2.28.

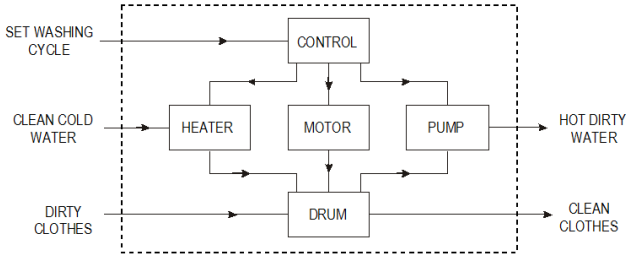


Figure 2.28 System boundary diagram (SGTS, 2012)

2.4.9.1.3 Design structure matrix

Alternately, system (or graphical) boundary diagram can be represented in a matrix based approach, often known as design structure matrix, as shown in Figure 2.29. It is a structured tool that visually represents the relations among system elements (modules or subsystems or components) (Steward, 1981; Tang et al., 2010; Malmqvist, 2002; Danilovic & Browning 2007).

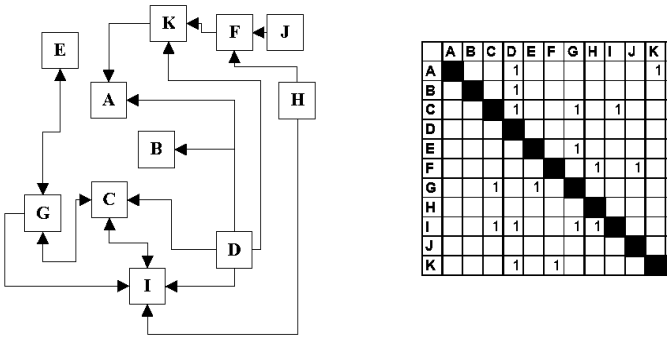


Figure 2.29 Design structure matrix and corresponding directed graph (adapted from Dong, 2002)

Design structure matrix, in literature, also referred as intra-domain and n-squared diagram (Tomiyama et al. 2009; DoD, 2001). Figure 2.29 shows DSM representation from a directed graph. Each link in the graph among elements interaction is depicted by '1' in DSM in the off-diagonal square boxes.

2.4.9.2 Summary

This section has discussed the representation of system's external and internal structure view. The interactions or dependencies between external and internal entities can be identified via graphical and matrix based approaches. The following table summarizes the types and viewpoints of structures that belong to any system-of-interest.

Table 2.10 Types and viewpoints of system structure-[S]

References	Viewpoint		Definition
Kossiakoff et al., 2011	External Actors	Stakeholders, & Environment	*"External Entities. These constitute all entities in which the system will interact. Many of these entities can be considered as sources for inputs into the system and destinations of outputs from the system"*
Eisenbart, 2014	Internal Actors	Sub-systems	*"Technical sub-systems encompass technical systems (i.e. technical products, potentially combining mechanical, electrical, and software systems with associates services), which are part of the system under consideration".*
References	**Types**		**Definition**
Dictionary.com, 2016	External Structure		*"of or relating to the outside or outer part; outer":*
Reverso Dictionary, 2016	Internal Structure		*"situated within, affecting, or relating to the inside of the body"*

The external structure of a system constitute entities in which system interacts and thus can be regarded equivalent to black-box view. The internal structure constitutes entities inside the system and thus can be regarded equivalent to white-box view.

In the next section, two aspects are discussed in detail. First, it is seen how a system structure and its interface information is perceived in its working environment (i.e. external structure - interacting actors outside the system) as well as between its subsystems (i.e. internal structure). Secondly, how and what viewpoints are considered on system interfaces both internally (white-box view) and externally (black-box view).

2.4.10 Interface view on a system

Interface definition and analysis are essential activities and integral parts of systems engineering process. During early design phases, emphasis is on ensuring that interface requirements are anticipated and coordinated carefully among different engineering departments responsible for designing separate systems (Lalli et al. 1997; Blyler 2004; Rahmani & Thomson 2012). An interface requirement addresses the functional, physical, environmental, and human requirements with specifications or characteristics that can occur at a shared boundary between two or more subsystems, configuration items, or systems (NASA, 2007).

There are two key reasons to look at system interfaces. “One is to analyse *product architecture* and the other is to *ensure compatibility* of subsystems in product development” (Rahmani, 2012).

2.4.10.1 Interface definition

The term *interface* is generally used to denote the shared boundary between two systems facing each other (Miller & Elgard 1998; NASA 2007; Otto & Wood 2001). In design literature, researchers define and conceptualise an interface in numerous ways such as a plane or place (Grady, 2006), a logical or physical relationship (Rahmani & Thomson, 2012), an internal feature (Gedell et al., 2011), an intended interaction location (Liang & Paredis, 2004), a spatial region (Wie et al., 2001), a linkage (Mikkola, 2001), a mating face (Blackenfelt, 2000) and also in other ways in (Fosse & Delp, 2013; Sellgren & Anderson, 2005; Sellgren, 1998; Ullman, 2010). In interface modelling, the first step is often to

define the boundary of the system which requires the identification of the interacting actors in its environment that can be human, supporting systems, and natural environment (Kossiakoff et al., 2011; Lalli et al., 1997). An interface can involve a contact as well as non-contact *interactions* between two systems (Ulrich, 1995). Table 2.11 summarises key definitions and perceptions of researchers for an interface. The different perceptions on interfaces of various researchers are reviewed, presented and discussed in detail by Parslov & Mortensen (2015).

Table 2.11 Definition of an interface

Source	Definition of an Interface
Ulrich, 1995	*"Interfaces may involve geometric connections between two components, as with a gear on a shaft, or may involve non-contact interactions, as with the infrared communication link between a remote control and a television set".*
Lalli & Kastner, 1997	*"An interface is that design feature of a piece of equipment that affects the design feature of another piece of equipment".*
Baldwin & Clark 2000	*"Interfaces are detailed descriptions of how the different modules will interact, including how they will fit together, connect, communicate and so forth".*
Otto & Wood, 2001	*"Interfaces are the boundaries between clusters, four types of interactions exist as flows across interfaces: spatial (geometry), energy, information, and material interactions. These interactions represent what must be shared across interfaces within the performance requirements".*
Mikkola, 2001	*"Interfaces are linkages shared among components, modules, sub-systems of a given product architecture".*
DoD, 2001	*"An interface is a functional, physical, electrical, electronic, mechanical, hydraulic, pneumatic, optical, software, or similar characteristic required to exist at a common boundary between two or more systems, products, or components".*
Albers et, al, 2004	*"Working surface pairs are all pair-wise interfaces between a component and its environment. This can be solid surfaces of bodies or boundaries with surfaces of liquids, gases or fields which are in permanent or occasional contact with the Working Surface. They take part in the exchange of energy, material and information within the technical system".*
Wie et al, 2004	*"A spatial region where energy and/or material flow between components or between a component and the external environment".*
Chen & Liu 2005	*"Interfaces possess the interacting functions such as connecting, transferring, transforming, and controlling. Physical interface specifications define the interacting protocol between components, and the geometric matching of existing physical connections".*
Scalice et al. 2008	*"An interface is an area where there is a flow of energy, material, information or at least, a spatial interaction among two or more modules or parts".*
Rahmani & Thomson, 2012	*"An interface refers to any logical or physical relationship required to integrate the boundaries between systems or between systems and their environment".*
Fosse & Delp, 2012	*"The system boundary that is presented by a system for interaction with other systems".*
Grady, 2006	*"An interface is a plane or place at which independent systems or components thereof meet and act or communicate with each other".*

2.4.10.2 Types of an interface

Interfaces have been categorised on the basis of a system's boundary, external interacting actors' types as well as on the basis of internal subsystems categories (e.g. hardware, or software etc.) as summarised in Table 2.12. The tables reflects the fact that researchers define and use interface meaning in slightly different ways.

Table 2.12 Types of interfaces

Reference	Interface Types	Definition
Boundary's logic based interface types		
Chen & Liu, 2005	External Interface	*"External interfaces connect external products (e.g. complementary products) or users and affect the upper level technical systemic performance".*
Sage & Lynch, 1998		*"An external interface to an element will have at least one internal terminal and one external terminal. The media connecting the terminals will breach the plane and separating the internal from the external terminal through a connector. An external terminal can be an environment".*
Chen & Liu, 2005	Internal Interface	*"Internal interfaces coordinate functional elements to perform full product functions".*
Sage & Lynch, 1998		*"An internal interface is one that resides inside the element of focus, including its terminals, connectors, and media. A component internal interface can be and external interface to a subcomponent internal to the component".*
External interacting actors' based interface types		
Sellgren & Anderson, 2005	Technical Interface	*"Technical interface – an intended interaction relation between a pair of technical functional surfaces in or on a technical system or in the environment".*
	Interactive Interface	*"Interactive interface – an intended interaction relation between an ergonomic or communicative functional surface on a technical system and a sensory feature of a real or generic human".*
Internal physicality based interface types		
Chen & Liu, 2005	Physical Interfaces	*"Physical interface specifications define the interacting protocol between components, & the geometric matching of existing physical connections".*
Lalli et al, 1997		*"Physical interfaces are used to define and control the mechanical features, characteristics, dimensions, and tolerances of one equipment design that affect the design of another subsystem. They also define force transmission requirements where a static or dynamic force exists".*
Sage & Lynch, 1998	Hardware Interfaces	*"Hardware to hardware interfaces are functional or structural and can exist to perform a service, transfer information, translate force, or provide structure and support. Hardware interfaces are physical in that they are real objects that touch the environment and media".*
Lalli et al, 1997	Software interfaces	*"A software interface defines the actions required when interfacing components that result from an interchange of information. A software interface may exist where there is no direct electrical interface or mechanical interface between two elements".*
		"Software interfaces are not as clearly characterized as hardware interfaces. They can be functional, informational, or environmental. The

Sage & Lynch, 1998		*functional nature of software interfaces deal with the passing of control elements. Information interfaces deal with data, while environmental interfaces are normally related to human or hardware interfaces".*
Sage & Lynch, 1998	Hardware to Software Interface	*Hardware receives and sends information to/from software through some type of transducer that changes an electrical or electromechanical signal into bits and vice-versa. These bits are trans- mitted through hardware media and stored in hardware such as magnetic media, memory, or a cache until it is made available to the software. Software application developers sometimes fail to recognize that hardware serves as the media in all software interfaces.*

2.4.11 Interface modelling approaches

Interfaces are conceptualised in various ways by researchers. This section discusses in detail, the key diverse theories for categorising and perceiving interactions and interface in a different manner.

2.4.11.1 Conceptualisation of interface via different theories

2.4.11.1.1 Contact & channel model

Albers & Matthiesen (2002) introduced a unique approach for modelling a mechanical system based on its interactions with the external environment. The contact & channel model approach underpins the fact that system interacts with its surrounding elements via the physical contact of working surfaces to achieve its functions.

The theory is based on two key building blocks: 'working surface pair' and 'channel & support structure' that connect to working surface pairs. A single working surface pair describes an interaction between two directly connected working surfaces (e.g. pen_nib and paper) or pair-wise *interface* between system and its environment (e.g. pen and user-hand) or between two systems and also the *effects* that take place between them as shown in Figure 2.30. Working surface pair' can be solid surface with surface of gas or liquid or solid taking part in the interaction exchange of flows such as energy, material, and information (Albers et al., 2004). 'Channel & support structure' is an intermediate medium or common element (solids, liquids, gases or spaces containing fields), which indirectly connect at least two 'working surface pairs'. Therefore, a single function in this approach requires interpretation of at-least two WSPs and the connecting CSS (Albers & Matthiesen, 2002).

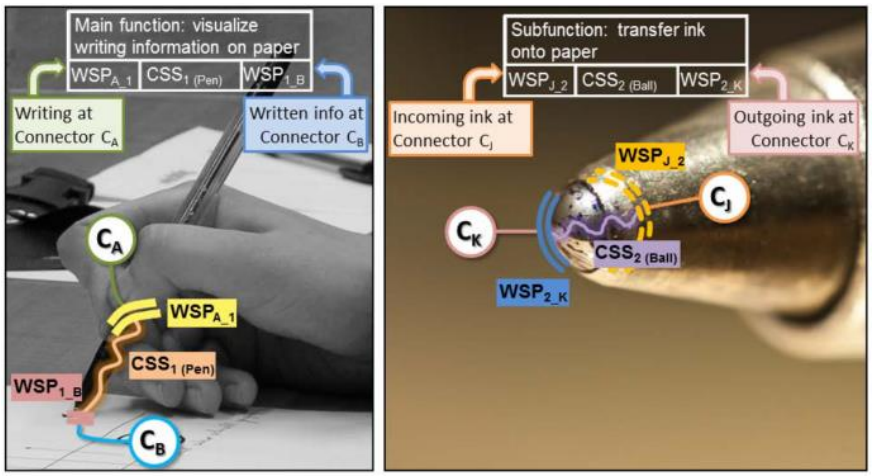


Figure 2.30 Contact & channel model descriptions of a ballpoint pen (Albers & Zingel, 2011)

'The theory has been expanded further with additional elements such as connectors considered necessary for fully embedding the system model into the environment (Albers & Zingel, 2011). The contact & channel model theory can model any technical system at any level of detail due to development and visualisation of both functional and physical elements simultaneously (Devanathan et al., 2009; Albers et al., 2004; Albers et al., 2009). However, the practice on this theory is limited to the analysis of existing technical systems as highlighted by Eckert et al. (2010). The following key viewpoint is drawn from this theory:

- An interface exists at working surface pair where two systems interact.

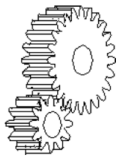
2.4.11.1.2 Theory-of-affordance

Maier & Fadel (2003) introduced a theory that describes technical artefact (or system) interactions from its 'potential uses' perspective with its surrounding interfaces and named it as affordances. According to them**,** *affordance* is "a relationship between two subsystems in which a *potential behaviour* can occur that would not be possible with either subsystem in isolation" (Maier & Fadel, 2009).

The theory is based on two types of interaction groups. Interactions between users and artefact are called as 'artefact-user-affordances' and the interactions between two artefact subsystems as 'artefact-artefact-affordances'. Maier & Fadel (2009) also discuss that in a design of a technical artefact, artefact-

artefact affordances (i.e. subsystems interface) usually exist at a lower level to serve higher level ‘artefact-user-affordances’ (i.e. user-system interface) as illustrated in Figure 2.31. They further describe that in each interaction group, there can be positive and negative types of affordances. The interaction description is based on affordance language. Positive affordance means desired behaviour is possible e.g. ‘typeability’ as ‘artefact-user-affordance’ in the typewriter device, and negative affordance means undesired behaviour is possible i.e. ‘noiseability’ or ‘annoying noise to user’. Positive ‘artefact-artefact-affordance’ in the gears, is ‘turnability’ and negative is ‘wearability’ or ‘wear’. All these describe potential behaviours of a device both desired and undesired in relation to its internal and external environments. The design process and the various properties of this theory are discussed in detail in (Maier and Fadel, 2009). One key property is that of *form dependency*. According to Maier & Fadel (2003) “a crucial difference between functions and affordances are that functions are form independent whereas affordances are form dependent”. The following key viewpoint is drawn from theory of affordance:

- Interactions can be positive and negative.



Design Level	Interaction Relationship Groups	Types of Interaction Positive (Improve) / Negative (Avoid)	Example
Higher Level	Artifact-User-Affordance AUA	(+) AUA Typeability	Typewriter
		(-) AUA Annoying noise	
Lower Level	Artifact-Artifact Affordance AAA	(+) AUA Turnability	Gears
		(-) AUA Wearability	

Figure 2.31 Affordance based interactions description

2.4.11.1.3 Port-based ontology

Liang & Paredis (2004) describe modelling of a system via ports-based theory. They considered ports as conceptual locations of intended interaction between two subsystems or a subsystem and its environment. To represent the necessary interaction information between two interacting subsystems, they proposed structured ontology by characterising the port attributes into function, form and behaviour concepts. Cao et al. (2013) also used and elaborated port-based modelling concepts for conceptual design.

Coherent with Liang & Paredis (2004) port based ontology concepts, Rahmani (2012) developed an automated (computer aided) interface definition and

control methodology but based on port-ontology. The key features of Rahmani's approach, in Figure 2.32, is to assess the compatibility of two subsystems and to highlight conflicts resulting from change of values of shared attributes (i.e. performance requirements) related to energy, material and information on either side of an interface using mating and controlling grammar rules. However, their approach seems restricted to hardware interfaces and white-box view as interface connection between two systems is perceived as ports which prompts that one should have knowledge about subsystems' ports. There are applications where there is no physical/hardware ports such as integrated modular avionics (AEEC, 1997). The following key viewpoint is drawn from Rahmani's ontological model:

- Shared measurable attributes related to energy, material, information, and geometry need to define between subsystems at white-box.

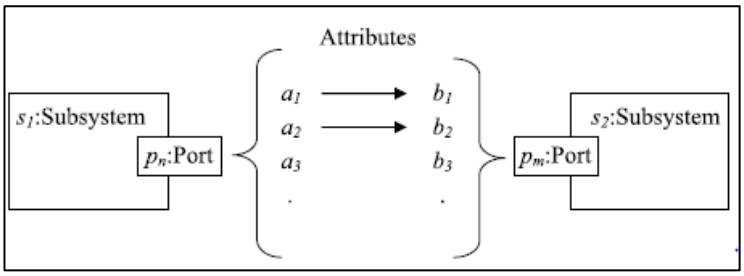


Figure 2.32 Port based ontology for subsystems interfaces (adapted from Rahmani, 2012)

2.4.11.2 Interface definition & documentation approaches

In previous section, analysis and conceptualisation of interfaces via different theories have been discussed. In this section, it is explored in detail how interfaces and in there interactions are defined and documented via which sets of tools.

2.4.11.2.1 Interface definition approach

To specify or record interface specifications of a system or subsystem, mostly tabular templates are suggested with different names in literature. For example, interface requirement document (IRD) (Grady, 2006) or interface requirement specification (IRS) or interface control document (ICD) (Lalli et al., 1997) is

often used. Lalli et al (1996) and Rahmani (2012) provide following structured steps to define and analyse interfaces;

1. *Identify interfaces*; deals with who is going to interact with whom.
2. *Categorise interfaces*; deals with what sort of interfaces; either disciplines specific (e.g. mechanical, electrical etc.) or/and exchanges specific (e.g. energy, material, information or spatial).
3. *Document interfaces*; deals with documentation of data captured in first two steps.
4. *Analyse for compatibility analysis***;** deals with monitoring and verifying that defined and designed interfaces are compatible even after changes.

Since, this thesis primarily focuses on definition and detail analysis of system and subsystems interfaces, therefore the focus of this research is till step 3 of ICDs. These steps of interface definition are also aligned with Grady's (2006) interface definition steps.

2.4.11.2.2 Interface documentation approaches

2.4.11.2.2.1 Use case - events based interactions

Eriksson et al (2006) used tabular template and language notation of rational unified process – systems engineering (RUP-SE) (Rational Software, 2003), illustrated in Figure 2.33. According to Eriksson et al., the RUP tabular template forces the analysts to always think about system-actor *interface* in a structured manner, which is a key success factor for maintaining focus in the modelling of complex systems. Eriksson et al apply this template both at system black-box and white-box views.

The tabular template has got two key merits over traditional use case scenario template. First, the tabular template focuses on the actor-system interactions (at black-box) thereby characterising them into actor actions (i.e. input from actor to system) and system responses (output from system to actor) in separate columns. Secondly, the table also provides budgeted (i.e. a performance) requirement column to capture performance targets which is not discussed in similar way in the other traditional use case tabular templates (as discussed in section 2.4.3.2). The subsystem requirements (using traditional 'shall' format) are derived based on white-box actions and responses, shown in Figure 2.33.

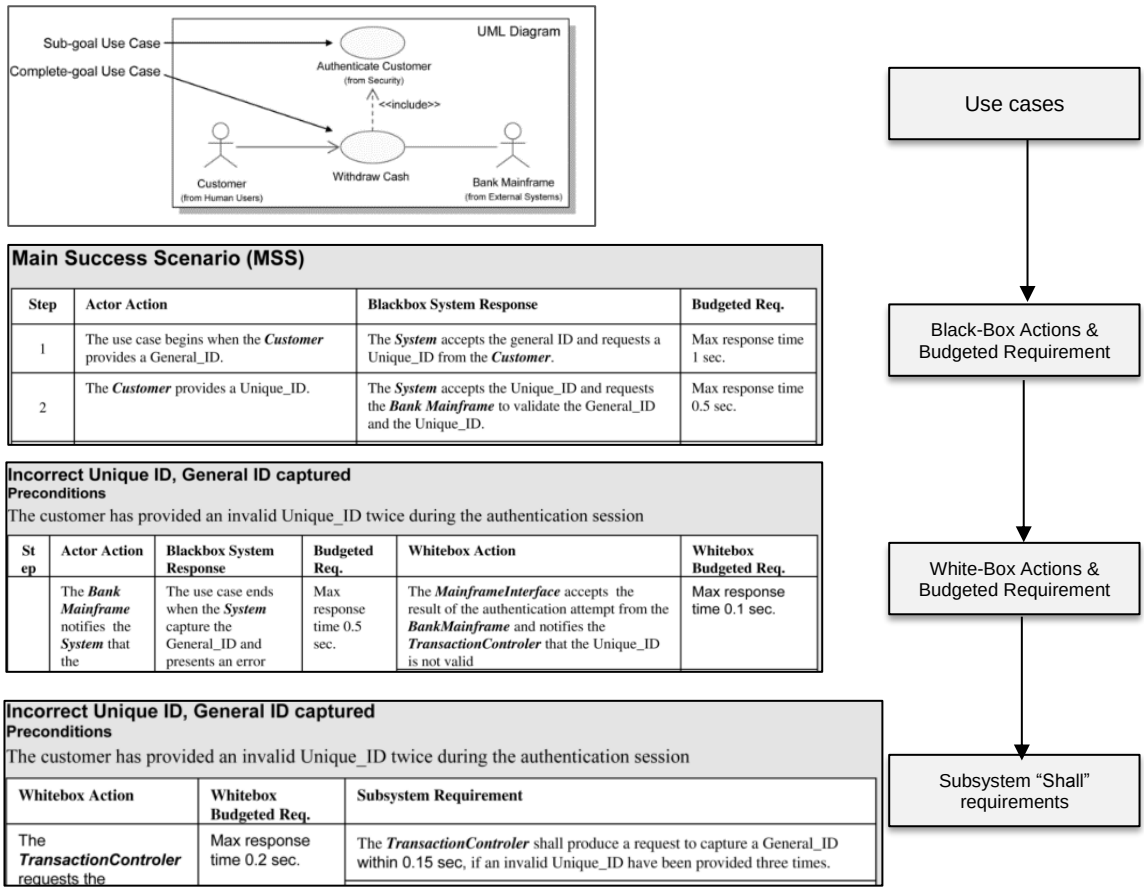


Figure 2.33 Use case diagram in conjunction with RUP-SE tabular template (adapted from Eriksson et al., 2008)

However, it is observed that only time related budgeted requirements are discussed in the tabular template whereas it is important to manage other (non-function) performance requirements as observed in Sections 2.4.3.1-to-2 and also related to energy, material and information related attributes as discussed in section 2.4.11.1.3. Following key viewpoints are extracted from this section:

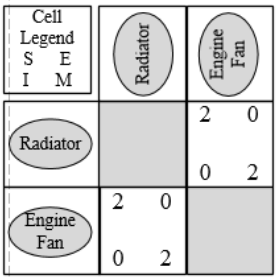
- Use cases (goals) are accomplished via interactions between external actors and system-of-interest interfaces;
- Interactions can be actions and responses based in scenarios;
- Budgeted & performance requirements need to be specified with interactions both at black-box and white-box views.

2.4.11.2.2.2 Flows based interactions

In a traditional design structure matrix approach a relationship between two components is described via a single qualitative *scheme* (discussed in section 2.4.9.1.3). However, such type of scheme does not describe multiple

interactions as realized by researchers (Pimmler & Eppinger 1994; Martin & Ishii, 2002; Jarratt, 2004) which is discussed in next section.

Interface analysis via matrix-based approach: The importance of design structure matrix based decomposition of a product and its integration analysis has been highlighted by Pimmler & Eppinger (1994). They used static design matrices to identify alternative product architectures and described four types of interactions referred as spatial (S), material (M), energy (E), and information (I) between the two interacting subsystems along with the quantification scheme that facilitated weighing interactions amongst them, as shown in Figure 2.34. Many researchers (Otto & Wood, 2001; Hamraz et al., 2013; Rahmani, 2012) have adopted and adapted (Helmer et al. 2010; Sosa et al. 2003; Blackenfelt 2001) this four-interaction taxonomy. For example, Blackenfelt (2001), looking at product variety and modularity in embodiment design, replaced the 'S' relation by inter-domain relation of 'FP' i.e. two entities contributing to the same function or parameter (FP). Sosa et al. (2003) extended the four-exchange taxonomy with an introduction of fifth type as 'structural' that indicated the requirements related to transferring loads or containment between two interfacing entities. However, in such approaches, one common problem is that in what sequence flows are identified i.e. energy first or material first?



Required	+2	Interaction necessary effect for functionality
Desired	+1	Interaction beneficial effect but not essential for functionality
Indifferent	0	No effect
Undesired	-1	Interaction causes negative effects but does not prevent functionality
Detrimental	-2	Interaction effect must be prevented for functionality

Figure 2.34 An excerpt of design structure matrix (adapted from Pimmler & Eppinger, 1994; Uddin et al., 2016)

One of the crucial decisions in the interface definition and analysis is to assess or prioritise and quantify the *interactions* that are normally considered with interaction exchanges between two subsystems (Pimmler & Eppinger, 1994; Campean et al, 2011). It is an important consideration as some interaction exchanges are more important than others e.g. 'rotational energy' transmission than 'thermal energy' for a shaft. According to Pimmler & Eppinger (1994) and

Campean et al (2011) interaction exchanges can have desirable effects (i.e. positive effects necessary for functionality) or detrimental (i.e. causing negative effects thus may or may not affect system functionality). They use five-point scale scheme from -2 to +2 to highlight the interactions criticality as shown in Figure 2.34. Following key viewpoints are extracted from this section;

- Interaction exchange types: flow related (energy, material, and information) & form related (spatial or physical)
- Interaction exchange quantification: + 2 (=required) to - 2 (=detrimental)

Interface analysis via graphical approach:

Sequence diagram is often considered for interactions analysis between two systems or system and its surrounding actors as shown in Figure 2.35.

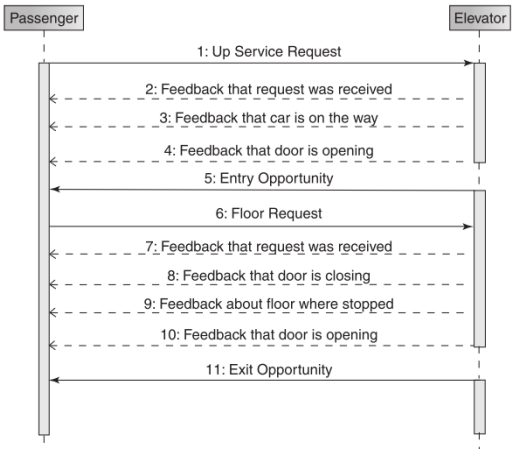


Figure 2.35 Interaction exchanges between passenger and elevator system (Buede, 2009)

According to Buede (2009), one critical issue in sequence diagram is that it is not clear what horizontal arrows should represent. Few researchers consider the horizontal arrows as sequence of operations (i.e. services) performed by the system-of-interest with its surrounding actors representing its interaction behaviours while others represent the flowing items (information, energy or material) being transferred from one system to another. Buede (2009) models the sequential interactions as flows or exchange of items on sequence diagram as shown in Figure 2.35. “Interactions involving the movement of data, horizontal arcs from the originating system to the receiving system, designate energy or matter among systems. A label is shown just above each arc to describe the data or item being conveyed” (Buede, 2009). A following key viewpoint is extracted from this section;

- Sequence of interaction exchanges between two systems can be imagined.

Interface analysis via tabular approach

Campean et al (2011) proposed interface analysis tabular template, in Figure 2.36, to detail the interactions in concrete manner thereby utilising the abstract information captured in matrix based approach (see Fig 2.36). Campean et al. apply tabular template at system's white-box and derive functional requirements (as verb-noun format in sixth column from left to right in Figure 2.36) based on identified interaction exchange descriptions (material, information, energy and physical related in fifth column) between two subsystems (i.e. internal interfaces) as well as between a system's internal subsystem and external actors (external to internal interfaces or vice versa) (i.e. user and environment). The key principle is that functional requirements need to be introduced in order to manage desired and undesired exchanges, and these functional requirements need to be (i) mapped against high level main function(s) of function structure (column seven in Figure 2.36) and (ii) cascaded through the system levels (Campean et al, 2013). However, there is no discussion of stakeholders' operational language as well as no discussion on performance related aspects and requirements. Following key viewpoints are drawn from this section;

- Interaction exchanges need to be identified between internal and external interfaces of a system;
- Functional requirements need to be specified with identified exchanges at interfaces;
- Functional requirements need to be linked with main functions.

Cell Ref	Interface	Type	Effect	Description	Function Required	High Level Function
1-B 2-A	Charger / Battery Pack Assembly	P				
		E	2	HV/HC from Charger to Battery pack	Transmit Electrical Power from ***Charger*** to ***Battery***	Charge Battery
		I	2	Battery Temperature info to charger	Detect Battery state of charge (SoC)	Charge Battery
					Transmit Battery state of charge info to Charger	Charge Battery
		M				
1-E1	Charger / Driver	P	1	Plug / Socket handling to connect to PIP	Provide ergonomic design of socket (Charger interface with Driver)	Charge Battery
		E	1	Mechanical, Driver access to socket	Allow Driver access to Charger socket	Charge Battery
			-1	Mechanical shock, vandalism	Design shock resistant / vandal proof charger / socket	Charge Battery
			-2	Electric shock to driver while charging	Isolate Driver electrically from mains power flow to Charger	Charge Battery
		I	1	Provide a visual indication that mains power is flowing to the charger	Indicate visually to Driver that electric connection to Charger is made	Charge Battery
		M	-1	Possible transfer of material (grease, paint, dust) from Driver to Socket or Socket to Driver	Prevent material exchange between Driver and Socket	Charge Battery

Figure 2.36 Interface analysis table (Campean et al, 2011)

2.4.11.2.3 Other tabular based templates for interface analysis

Apart from use case diagram, context diagram, textual interaction operations-based templates at black-box and also exchanges-based matrix, graphical & textual templates at white-box (Sections 2.4.9.1.1, 2.4.11.2.2.1, & 2.4.11.2.2.2), many other tabular ICD or IRS templates have been developed to cover the interfaces in sufficient detail between subsystems of system-of-interest and its surrounding actors, shown in Figure 2.37.

Ullman (2010) provides guidelines and a template as a reverse engineering method. In Ullman's reverse engineering approach, first step is to examine existing device's interfaces (e.g. Quick-grip device) with other interacting objects (e.g. user) and identification of energy, material, and information flowing in and out of it as shown in upper half of template in Figure 2.37a. In the second step, a component (e.g. Trigger) is chosen for in depth study to understand its role and behaviour to the rest of device. In the final third step, each interface is examined to identify the flows in detail (see lower half of template).

Otto & Wood (2001) also emphasizes about identification of flows from system's black-box top-function to sub-function till modules. Once modules (e.g. actuation, electrical system and base), in Figure 2.37b, are identified, their interfaces with each other are listed along with four types of interactions in and out in each interface.

Interfaces with other objects:

Part #	Part Name	Other Object	Energy Flow	Information Flow	Material Flow
1 & 2	Main body and Trigger	User's hand	User squeezes trigger to move jaws closer together and	Squeezing force proportional to jaw force	User's hand grips and releases
8	Pad	Parts being clamped	Clamping force and compressive motion of jaws moving together	None	Parts flow into and out of jaws
Etc.					

Flow of energy, information, and materials:

Part #	Part Name	Interface Part #	Flow of Energy, Information, and Material	Image
1	Trigger	User	Force 1a applied by gripping trigger and main body. Resistance force felt by user proportional to clamping force.	

(a) Reverse engineering template for Quick-grip (adapted from Ullman, 2010)

		Interaction Type				Interfaces In: Modules (External Systems)		Interfaces Out: Modules (External Systems)
Module	**Interactions**	**M**	**E**	**S**	**Sp**		**Sp**	
Actuation	Electricity		•			Electrical system		Inside base
	Human force		•		•	Human hand		
Inside base	Inside air	•			•	Passenger compartment	•	Inside heater, inside cooling
	Electricity		•			Actuation	•	Inside heater, inside cooling

(b) Modules listing with interfaces and interactions (adapted from Otto & Wood, 2001)

INTERFACE IDENTIFICATION		INTERFACE CONNECTIVITY		
ID	NAME	SOURCE	DEST	MEDIA
I1511	Fuel On Command	A21	A252	A32

MID	MODEL ENTITY	PID	PRODUCT ENTITY	RID	REQUIREMENT
I127	Radar Altitude	A21	Radar Altimeter	G74H	0 ≤ Altitude ≤ 5000 feet
I127	Radar Altitude	A22	On-Board Processor	QJU7	0 ≤ Altitude ≤ 5000 feet

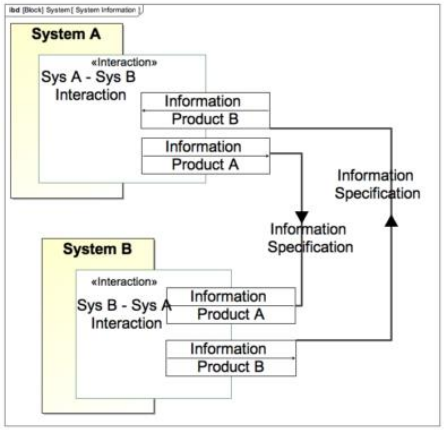
(c) Interface dictionary and requirements allocation sheet to capture interface requirements (adapted from Grady, 2006)

Source	Target	Interface / Flow (Voice of the Customer)	Interface Category & Effect	Description (Voice of the Engineer)	Range	Unit	Other information	
							Source Owner	Target Owner
Between **IP Assy – Complete** and **Windscreen**								
A1	B3	IP assy emits demist air onto windscreen	M-	Volume of air emitted	x-y	lit/min	IP Engineer	Glazing Engineer
B3	A1	Windscreen provides viewing window for VIN plate	P+	Position variation of obs band window around c/l of VIN plate	Xyz max	mm	IP Engineer	Glazing Engineer
Between **IP Assy – Complete** and **Carpet Assy**								
…								

(d) Interface analysis worksheet (adapted from JLR, 2014)

From	To	Interface / Flow	Interface Category	Range	Unit	Description	Technical Specification	Importance	Impact	Interface Partner
Between **(7) Vehicle Driver – (4) PDA**										
(7)	(4)	Brightness of display	E,I	150 - 200	cd/m^2	Adjustment of brightness for night & day	DIN EN ISO 9241	high	+	Supplier of touch screen
…	…	…	…	…	…	…	…	…	…	…

(e) Interface description sheet (adapted from Fritzsche, 2008)



Interaction Specification	Inputs	Outputs
System A (Multi-Mission Operation System) – **System B** (Flight System) **Interaction**		
Interface Operation: Update Commands	Cmd	Telem
Interface Operation: Update Configuration	Cfg Loads	Cfg Telem
System B (Flight System) – **System A** (Multi-Mission Operation System) **Interaction**		
N/A		

(f) System interaction information (adapted from Fosse & Delp, 2012)

Interface Type	Interface From…	Interfaces To…	Description	Other information
Application Layer				
<interface types>	*<interface from>*	*<interface to>*	*<enter description of interface>*	*<other supporting information>*
<interface types>	*<interface from>*	*<interface to>*	*<enter description of interface>*	*<other supporting information>*

(g) An interface tabular template by CMS (2013)

Figure 2.37 Tabular templates for capturing interface requirements

According to Grady (2006), every interface of a system contains three aspects i.e. source, destination (i.e. technical solutions) and a media (wires, bolts etc. i.e. more like physical interface) between source and destination solutions. This can be captured using typical interface dictionary listing as shown in Figure 2.37c. Grady recommends using *requirements analysis sheet* to capture any type of requirement including interface requirements. For example, in Figure 2.37c, a model entity (e.g. radar altitude) is the common interface (i.e. exchange) between the implementations i.e. source (A21) and destination (A22) elements with shared measurable performance requirement (e.g. $0 \leq \text{altitude} \leq 5000$).

Automotive industry often uses interface analysis sheet which seems to be a combination of viewpoints/contents highlighted and used by Otto & Wood (2001), Grady (2006), and Ullman (2010). However, one key difference is the articulation of interface description via two mind-sets i.e. ‘voice of customer’ and ‘voice of engineer’ as shown in Figures 2.37d & e which is not seen in academia research templates. The impact/effect of an interaction type is also assessed which in principle is analogous to Pimmler & Eppinger’s (1994) approach. However, functional articulation language is used quite weakly in the column of voice of engineer and is often used interchangeably with customer language or interface exchange specifications. Furthermore, whether interactions in and out from a subsystem or module are also not clear as seen clearly in Otto & Wood (2001) and in Fosse & Delp (2013) interaction analysis model.

Fosse & Delp (2012) describe robustly and distinctly interaction specifications and interface functions between two systems via model based approach. Interface properties related to information/material in and out (e.g. Cmd and Cfg loads) are regarded as equivalent to ports in SysML (system modelling

language) and interface functions (e.g. Update Commands) equivalent to interface operations in SysML as shown in Figure 2.37f.

There are also other interface analysis oriented approaches available in literature (see e.g. Cabigiosu et al., 2013), however those are also limited to specific discipline such as centre for medicare and medicaid services (CMS, 2013), aircraft/stores (Schlatt, 2004) and provide limited information (see Figure 2.39g). Furthermore, other interface data models have been designed such as for physical interface ontology to explore the possible physical interfaces (i.e. media related) and their conflicts between two technical solutions, see e.g. Holley et al. (2014).

Following key viewpoints are derived from the above templates;

- Interface analysis should describe clearly ‘voice of the customer’ (i.e. non-technical or natural language) and ‘voice of the engineer’ (i.e. technical);
- Interactions directionality should be specified with (i) interacting actors i.e. from whom to where as well as (ii) for flows/exchanges in and out of a system at black-box or for subsystem at white-box.

2.4.11.3 Summary

The *interaction information* between interfacing systems needs to be sufficiently detailed to communicate meaning with minimal or no ambiguity (Kossiakoff et al., 2011; Lalli et al., 1997). It is seen that *interface requirements* (i.e. statement) are normally made (or derived) in relation to the interaction operational actions, functional or logical (including physical characteristics) OR exchanges based relationships that are required to exist at a system’s external as well as within its internal boundary with its operating environment. The variation and diversity in interaction viewpoints show clearly that all these viewpoints are essential and carry specific meaning and purpose while defining interfaces and conducting interface analysis. Table 2.13 summarises all these viewpoints.

Table 2.13 Consideration of viewpoints within **i**nterface view-[I]

Reference	Viewpoint		Definition
Malan & Bredemeyer, 2001	Goal (use case)		*"A use case defines a goal-oriented set of interactions between external actors and the system under consideration... use cases capture who (actor) does what (interaction) with the system, for what purpose (goal) without dealing with system internals".*
Eriksson et al. 2008	Interaction (as operation, actions & responses)		*"… user goals are further specified by a number of scenarios describing the interaction between a system and its actors (users and environment) with the purpose of achieving these goals."*
Campean et al. 2011;	Interaction (as exchange)	Flows (E, M, I) related	*"… interactions could involve exchanges of energy / material / information, and could affect the functional performance of the system. Therefore, these interfaces need to be identified, characterized (i.e. define the type of exchange … and managed through functional requirements."*
Pimmler & Eppiner, 1994			*"An energy-type interaction identifies needs for energy transfer between two elements". "An information-type interaction identifies needs for information or signal exchange between two elements". "A material-type interaction identifies needs for material exchange between two elements"*
Pimmler & Eppiner, 1994;		Form (P) related	*"A spatial-type interaction identifies needs for adjacency or orientation between two elements"*
Campean et al. 2011;			*"…it is useful for the engineering interface analysis also to identify, analyse and specify any Physical Contact requirements at an interface…. Physical contact usually provides a path for energy / material / information exchange, and therefore it is commonly used as a first point in interface identification and analysis."*
Kossiakoff et al., 2011	Interaction directionality	Interaction In/Out	*"…the interactions between the external entities and the system and are represented by arrows. Arrowheads represent the direction or flow of a particular interaction".*
		From/to	
Pimmler & Eppiner, 1994; Campean et al. 2011	Interaction (effect) relation	Desired	*"Some interactions are described as desirable"*
		Undesired	*"…while others are detrimental"*
Pimmler & Eppiner, 1994; Campean et al. 2011;	Interaction (effect) quantification	Five point scale	*"Interactions can be quantified on a five point scale (-2,-1,0,+1,+2) based on the relative need for each interaction type".*
NASA, 2007	Interaction Requirement (Specification)		*"An interface requirement defines the functional, performance, electrical, environmental, human, and physical requirements and constraints that exist at a common boundary between two or more functions, system elements, configuration items, or systems".*
Fosse and Delp, 2012			*"Describes how an operational entity (system, organization, or service) can effect another operational entity when a connection exists".*
Campean et al., 2011		Functional	*"Specification of functional requirements in 'verb-noun' format in relation to exchanges".*
Grady, 2006		Non-Functional (Performa	*"Specifies the performance aspects with constraint or bounding relations (<, >, = or min / max), target value, and unit associated with functional requirement".*

		nce related)	

2.4.12 Descriptive reasoning models on a system

The models have been developed to reason about the relationships and engineering descriptions among the conceptualised views of a technical system. Two key models in engineering design are discussed in this regard.

2.4.12.1 Four concepts-based descriptive model

Brown & Blessing (2005) have introduced descriptive reasoning model based on four key concepts in engineering descriptions of technical devices or systems. These four concepts are goals, operations, functions, and behaviour. Firstly, goals describe the usage of agent with the technical system in the desired state of the world. Secondly, operations describe the sequence of events in a plan in order to achieve a goal. Third, functions describe the roles, a technical system plays in its environment, when agent uses the system. Fourth, behaviours are the result of causal interactions between device and environment that can be values of state variables or relationships between them at an instant or over time. Brown & Blessing deployed their model to analyse the concept of affordance theory in terms of goals, operations and functions. They conclude their detailed model based on ball-point pen example that contains many other interconnected viewpoints among these four concepts which is summarised in Table 2.14. Brown & Blessing model emphasise less on structure concept which is introduced and elaborated by Vermaas (2009) in his five concepts based reasoning model which is discussed next.

Table 2.14 Technical system's description model by Brown & Blessing

A complete description model for a function of a device: {D,M,R,B,C,O,P,I,G}	
D: Device	O: Operations
M: Mode of deployment (between D & environment actors)	P: Plan
R: Relationships	I: Intention
B: Behavioural constraints	G: Goal
C: Conditions	

2.4.12.2 Five-concepts based descriptive model

Perhaps, the most influential descriptive reasoning model is introduced by Vermaas (2009), in Figure 2.38, thereby accumulating and describing five key concepts or views of engineering descriptions of a technical system in engineering design. These five views are given names as *goals*, *actions (as operations/events)*, *functions*, *behaviours*, and *structures*. The definition of each is also given in Figure 2.38.

goal	a state of affairs desired to be realised by an agent with the device
action	a deliberate manipulation the agent is to carry out with the device in order to achieve the device's goal
function	an effect the behaviour of the device has to have for letting the device's actions be successful
behaviour	the evolution of the physical state of the device
structure	the physical configuration of the device in its environment

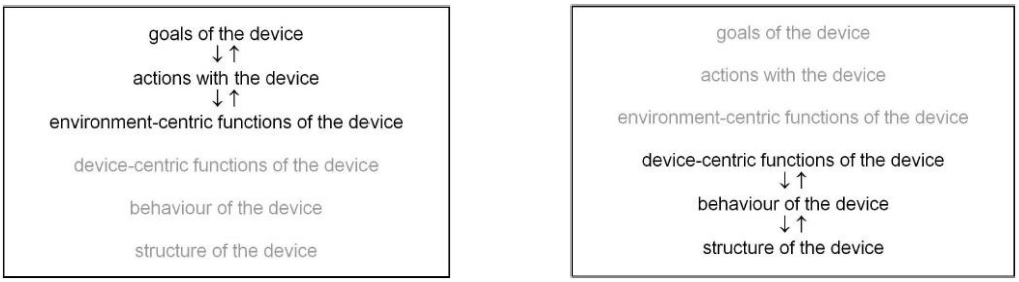


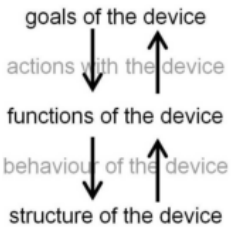
Figure 2.38 Conceptual views of a technical system (adapted from Vermaas, 2009; 2010)

The approach for describing or modelling a technical system can be perceived in two ways (Vermaas, 2009). The goals, actions, and functions are grouped that represent the designer (or agent) intentions with the technical system, calling it as *agentive or intentional perspective*. This is more like top-down approach starting with, and relating goals to actions till structures. The other way is bottom-up, calling it as *structural perspective* (or causality related) starting from structure and all the way upwards through these views.

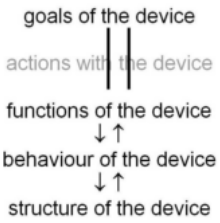
The reasoning process begins with the identification of the goals of the potential users and their associated possible actions (i.e. operations) with the device, in relation to specific use plan (i.e. *actions* to achieve a specific *goal*). In order to successfully execute this use plan, the analyst (or designer) have to derive the functions that the technical system should deliver along with the expected behaviour of system. Finally, the internal components (structure) need to be

explored in a way that the system as a whole should exhibit the expected behaviour and provide the desired functions.

(Vermaas, 2010) elaborates the concept of function via the device-centric and environment-centric functions thereby arguing that the design description of a technical system can be perceived thereby cloaking either the 'physics' (i.e. structural perspective) or the 'goals and actions' (i.e. intentional perspective) in device reasoning, as shown in Figure 2.38. Vermaas shows that due to such reasons (i.e. two different perspectives thinking), certain views are ignored and bypassed in existing functional modelling approaches, shown in Figure 2.39. This model does not explicitly discuss about interface concept in detail as well as performance requirement related aspect.



Functional basis (Stone & Wood, 2000) reasoning scope



Function behaviour structure (Gero, 1990) reasoning scope

Figure 2.39 Existing approaches' reasoning scope (from Vermaas, 2010)

2.5 Summary: An overview of literature review

This chapter has come across three key conceptualised characteristics on each of the five modelling views that are essential for system architecture analysis. These are (1) system modelling *viewpoints* (2) system modelling views' *types* and (3) system *decomposition views* on each hierarchical level. These three characteristics are accumulated and briefly discussed.

2.5.1 Viewpoints of system views

The literature review has helped in concluding the fact that a system's single view (e.g. requirement or function or structural view) is a collection of multiple viewpoints as summarised in Figure 2.40. The system viewpoints in each view, in Figure 2.40, have been aggregated on information in requirement view (Table 2.5), function view (Table 2.8), behaviour view (Table 2.9), structure view (Table 2.10) and interface view (Table 2.13). The viewpoints in turn contain many sub-

categories. These viewpoints help in modelling each view or deriving view's types robustly in a complete, correct and consistent manner.

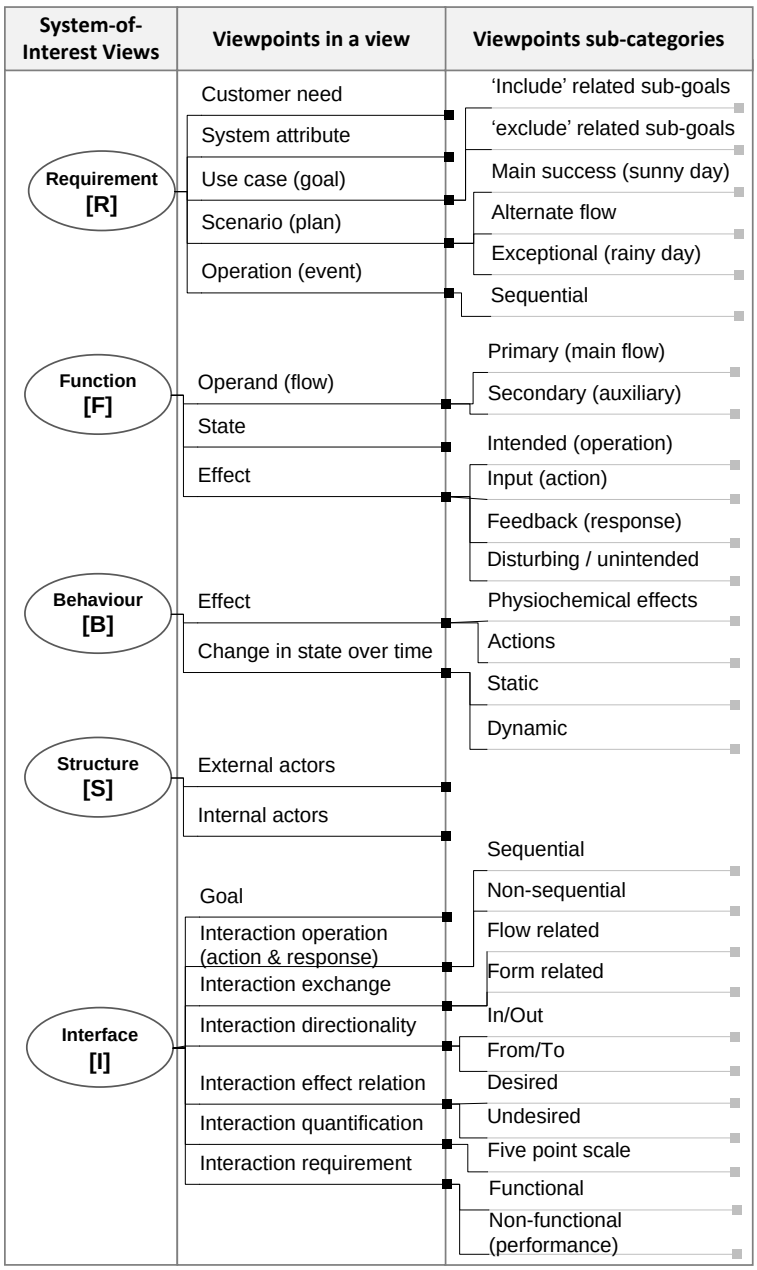


Figure 2.40 Taxonomy of system-of-interest modelling viewpoints

2.5.2 Types of system modelling views

The literature review also helped in concluding the views' types. In each view, classification is broadly twofold as summarised in Figure 2.41 discussed in relevant sections of each view. The main types in each view have been accumulated based on information in Table 2.4 (requirements types), Table 2.7 (functions types), Table 2.9 (behaviours types), Table 2.10 (structures types)

and Table 2.12 (interfaces types). For example, environment-centric & device-centric concepts within function view and functional & non-functional requirement concepts within requirement view. Each type may or may not have sub-categories such as functional requirements in requirement view can be input/output related as seen in Table 2.4.

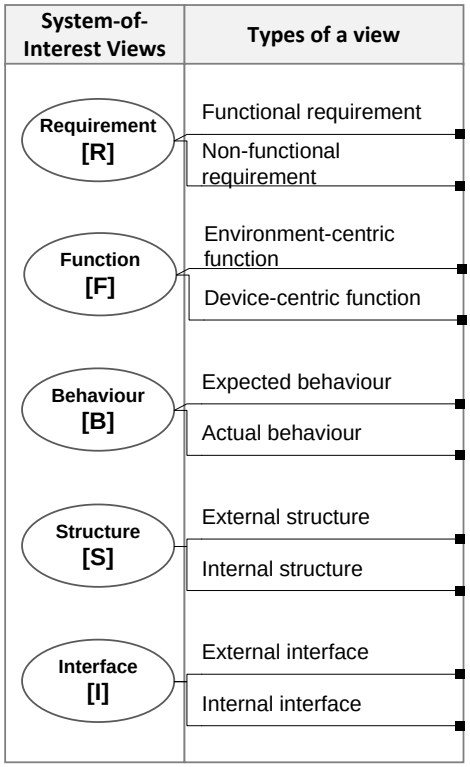


Figure 2.41 Taxonomy of modelling views' types

2.5.3 Types of decomposition views on a system level

A system-of-interest can be a subsystem itself (i.e. a white-box of a system) as well as it can be a system itself (i.e. black-box) that in turn contains many subsystems (at its white-box). The Figure 2.42 is derived based on review, in Figure 2.7, regarding system hierarchy representation.

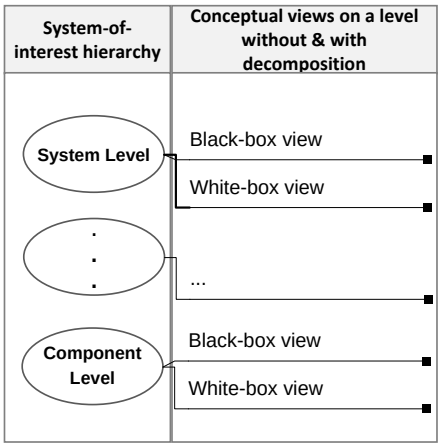


Figure 2.42 Taxonomy of conceptual system views on a hierarchical level

Therefore, a system-of-interest can be conceptualised from its black-box and white-box perspectives and it can be in any level of design hierarchy as illustrated in Figure 2.42.

2.6 Critique of existing approaches and theories

It is seen that different approaches and theories recommend several multiple viewpoints and conceptualised a system view from different perspectives. Section 2.5 gives a summary of three critical but essential characteristics that should be considered during modelling a system or analysing a system architecture.

The literature also provides an evidence that both graphical based (flow-charts and matrix) and tabular based tools are used by different theories and approaches for capturing a specific system view as well as linking the multiple system views, their types and viewpoints. Some describe the types and viewpoints in analysing a specific view. Others show traceability and linkage across the multiple views. These approaches differ in scope of views but support system architecture analysis activities and suggest different entry points for analysing a system architecture. It is quite difficult to compare these versatile approaches in their procedure and scope. This demands for a concrete reference model for the examination of existing approaches.

This section presents high level reflections on the reviewed theories and approaches both within engineering design and systems engineering communities with the aim of clarifying the broader gap in the existing research.

Key observations from literature review are articulated as follows:

1) Though the reviewed approaches provide different but correct ways of developing system architecture of a system, the design process models and proposed approaches lack to provide complete and consistent guidelines around the three key conceptualized characteristics as shown in Figures 2.40, 2.41, and 2.42.

2) There is no concrete reference architectural model that could unite these three conceptualized characteristics of a technical system.

3) System-of-systems thinking require consideration of these views' viewpoints, types, and conceptual views distinction on each hierarchical level.

These findings suggest that there is a gap of availability of concrete reference model at first place both in systems engineering and engineering design communities that should be encapsulating all the three conceptualized characteristics together due to which insufficient, incomplete, and several different guidelines are observed by the current design process models and approaches. Therefore, the following initial research hypothesis is established to direct the research:

*Integrating the multiple **views' viewpoints**, **types**, and the conceptual **views on a hierarchy** would provide a structured reference model for conceptualizing or modelling a technical system architecture in a robust manner.*

Figure 2.43 Preliminary research scope

2.7 Chapter summary

This chapter has reviewed the existing state of the art in the context of system design processes both in engineering design and systems engineering. Table 2.1 illustrates the fundamental views for a technical system architecture analysis which used in this thesis for literature review. The chapter has discussed and grouped the common and different conceptualised characteristics of system modelling approaches as well as researchers' perceptions around a technical system in tabular formats. The chapter has

concluded three conceptualised characteristics: modelling viewpoints, types of views, and conceptual box views on a design hierarchical level that are considered potentially useful for getting a correct, consistent and complete description of a technical system.

The next chapter initially develops the reference model based on which the limitations in existing modelling approaches are examined both from procedure and scope perspectives.

3. Research Methodology, Reference Model Development, and Evaluation of Existing Approaches

3.1 Introduction

This chapter initially develops the reference model as part of research methodology based on three key conceptualised characteristics associated with a technical system presented in Chapter 2. Existing modelling approaches are then evaluated based on the criteria drawn from the developed reference model. Many existing modelling approaches are found to be limited in their scope and procedure in the context of representing three key characteristics of a technical system. The critical research gaps and hypothesis are presented in this chapter. The overview of research methodology used in this research is also discussed in the end.

3.2 Development of the reference model

3.2.1 Purpose for the reference model

The need of a reference model has been discussed in Chapter 2 for the purpose of evaluating existing modelling approaches. The present research initially aims to unify diverse viewpoints for system architecture analysis onto a reference model. The reference model should also support the iteration and recurrence aspects for system-of-systems analysis. Iteration is repetition of steps at one hierarchical level whilst recurrence is repetition at next lower level (Bonnema, 2008). The selected design process and reasoning models both from systems engineering and engineering design communities are limited in such aspects and thus require combination and encapsulation of these aspects along with three conceptualised characteristics (Figures 2.40, 2.41 and 2.42) of a technical system.

3.2.2 Requirements for the reference model

Based on the critical review of the literature presented in Chapter 2, the following requirements on the reference model for system architecture analysis have been established:

- Specify clear names of modelling views (Table 2.1);

- Specify clear names of viewpoints that help in concretising a specific modelling view (Figure 2.40);
- Specify clear names of views' types (Figure 2.41);
- Reveal relevant types of modelling views at relevant conceptual box views (Figures 2.7 and 2.42);
- Allow for iteration (Section 2.3.4);
- Allow for recurrence (Section 2.3.4);
- Provide flexible structure to separate a specific view for in-depth study;
- Allow different paths for the development of novel methodologies not only within a single view but also across multiple views;
- Compatible with other descriptive reasoning models (Section 2.4.12)

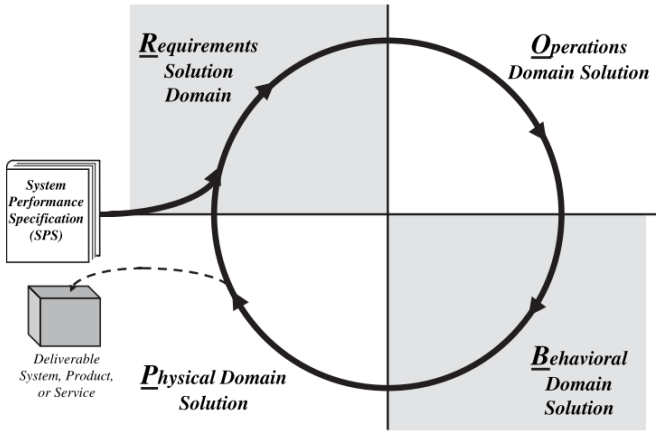
3.2.3 The reference model structure

Figure 3.1a shows a graphical structure of Wasson's model for a design process in which four key system modelling views (i.e. domains) are arranged: requirements, operational, behavioural, and physical. This model discusses iterative and recursive aspects among views. However, this model does not show the viewpoints in each view and also the types of different modelling views. Inspired by Wasson's model, this thesis develops a reference model and is represented via graphical structure in Figure 3.1b that arranges the modelling views in five sections and encapsulates the aforementioned requirements (as specified in Section 3.2.2).

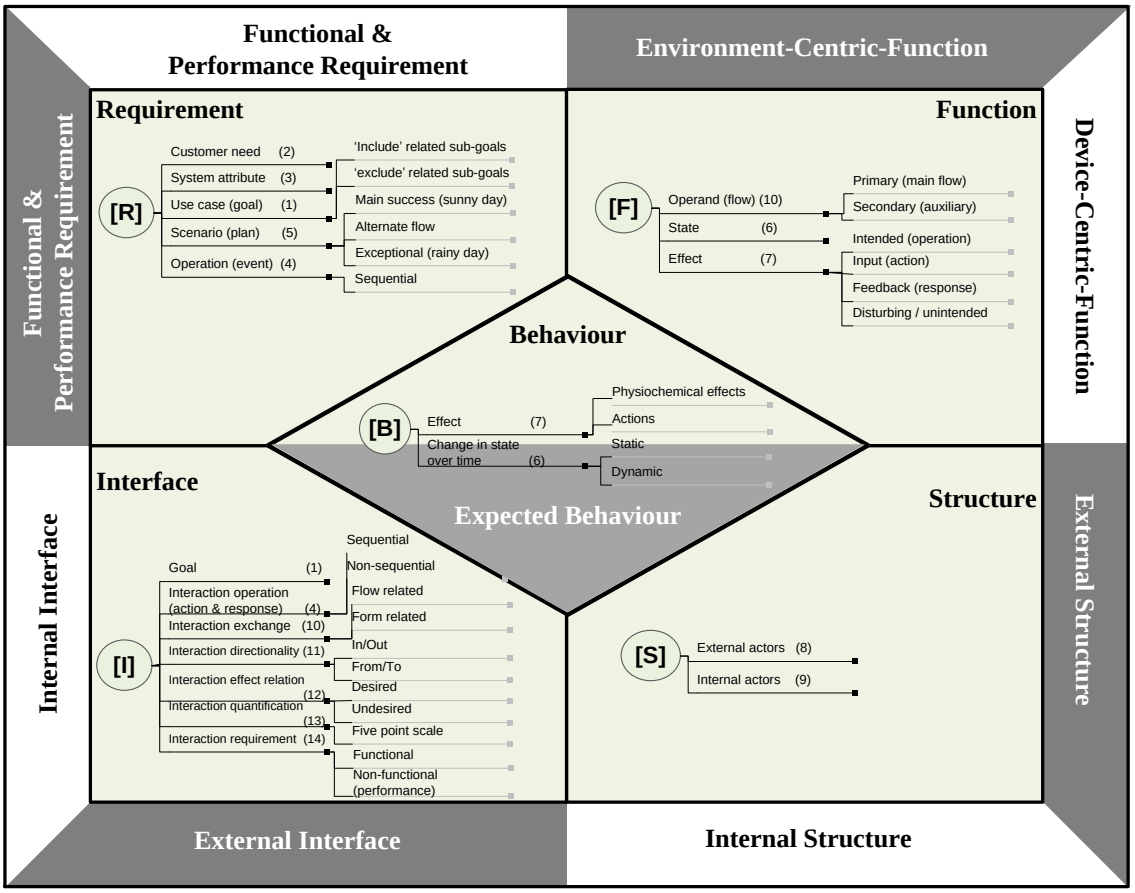
The inner square, in Figure 3.1b, shows views and viewpoints. The inner square is divided into five modelling views (Table 2.1) and in there the relevant viewpoints associated with each view are placed (Figure 2.40).

The outer square, in Figure 3.1, is partitioned with and without dark shaded regions. The two regions in outer square used for revealing system's conceptual box views on a hierarchical level (Figure 2.42): black-box view (dark shaded region) and white-box view (without dark shaded region). The different types of modelling views (Figure 2.41) associated with relevant black-box and white-box views are specified in the outer square. For example, in Figure 3.1, the 'environment centric function' type is specified on the dark shaded region revealing its definition and consideration at system's black-box view. Similarly,

'device centric function' type is specified on dark-less shaded region revealing its definition at system's white-box view. Both these types are specified next to 'function view [F]' located in the inner square of Figure 3.1. Same logic is used and applied for the types of other views.



(a) Wasson's (2006) system solution modelling via multi-domains iterations



(b) Developed reference model in this thesis

Figure 3.1 The reference model for conceptualising a technical system

Note that the reference model provides iteration at one hierarchical level and at same time it does not prescribe a certain route nor any specific entry point.

In the reference model, there are in total 19 viewpoints appearing on 5 views; however these are reduced to 14 viewpoints as 6 viewpoints are common across the several views, as listed in Table 3.1. The common viewpoints are given same numbers e.g. number '1' is assigned to 'goal' which is common across requirements and interface views as can be seen in Figure 3.1 and Table 3.1. Note the numbering of viewpoints in Figure 3.1 and Table 3.1 is same.

Table 3.1 Filtering common viewpoints across views

Viewpoints	**Common in views**	
1. *Goal* (= use case)	[R] , [I]	
2. *Need*		[R] related
3. *System Attribute*		
4. *Operation* (= event or action/response)	[R] , [I]	
5. *Scenario* (= plan)	[R] , [I]	
6. *State* (= change in state of operand or flow as well as change in state of subsystems over a period of time)	[F] , [B]	
7. *Effect* (1. actions or processes from [B] view = Intended inputs as actions & outputs as feedback**,** unintended i.e. disturbing inputs/outputs from [F] view) (2. = physio-chemical effects in relation to operands from [B] view)	[F] , [B]	
8. *External Actor*		[S] related
9. *Internal Actor*		
10. *(Interaction) Exchanges* (=flows/operands related)	[F] , [I]	
11. *Interaction Directionality*		[I] related
12. *Interaction Effect Relation*		
13. *Interaction Quantification/Scale*		
14. *Interaction Requirement (Functional & Non-functional performance related)*		

The 6 common viewpoints in Table 3.1 are used in similar meaning but slightly in different contexts in literature e.g. state viewpoint in [F] view is used in the context of 'exchanges' states transition whilst same viewpoint is used in the context of 'subsystems' states transition in [B] view. The operation viewpoint is

used in the context of events (or actions) occurring sequentially within scenarios between users or other stakeholders with the system so that a goal is realised by the system in [R] view. Similar thinking is observed in [I] view but mostly non-sequential based operational interactions.

It is also seen, on one hand, from Table 3.1 that interface view [I] is the most prominent one, followed by function view [F], in terms of viewpoints coverage belonging to other views.
These 14 viewpoints along with types of modelling views should be used as conceptual modelling elements for analysing a technical system architecture existing at any level of hierarchy. For example, an architecting model should allow designer (or engineer) to analyse different view's types on a system e.g. is it environment-centric or device-centric function of a system? Or in structure view; is it external structure or internal structure? by following black-box and white-box views.

3.2.4 Constraints on evaluation criteria for existing approaches

From the study perspective, it is observed, on one hand, some approaches are limited to a specific view but with detail procedure e.g. requirements derivation and interfaces definition. On the other hand, some approaches kept the study focus on traceability of information among multiple views but with minimal procedure for architecture analysis. Therefore, it would be inappropriate to use the reference model as it is in Figure 3.1 for examination of existing approaches due to variation in their study scope. Therefore, alternate graphical representation from the reference model will be extracted so that relevant approaches can be evaluated in the relevant scope of study. Note, one requirement on the reference model was 'to provide flexible structure to separate a specific view from others'.
The aim of criteria is to examine the approach's coverage/completeness as well as its procedural steps in the relevant field of study.

3.3 Establishment of evaluation criteria based on the reference model

3.3.1 From scope perspective

The reference model is split into two cubes oriented models as shown in Figure 3.2. These two cubes-oriented models would be used for evaluating the scope

of existing approaches. Note the numbering of viewpoints in cubes-models is same as in Table 3.1.

As seen and discussed (in section 3.2.3) that interface view contains most of the common viewpoints available in other views therefore it is divorced from rest of the views in Figure 3.2. It will not only support evaluating the approaches developed in this view but also the approaches that have same viewpoints too. At the same time, it is carefully considered that none of the viewpoint appears twice in these two cube models.

The C2-model is mainly a representation of interface analysis model comprising of interface view [I] with its viewpoints (see also Table 3.1). The interaction requirement (i.e. viewpoint no.14 in Table 3.1) can be functional and performance (non-functional) related therefore shown on two separate facets in C2-model as it will also support requirement view's types. The other viewpoints from no. 11-13 (in Table 3.1) of interface view are grouped in one facet as all these are associated with interaction exchange viewpoint.

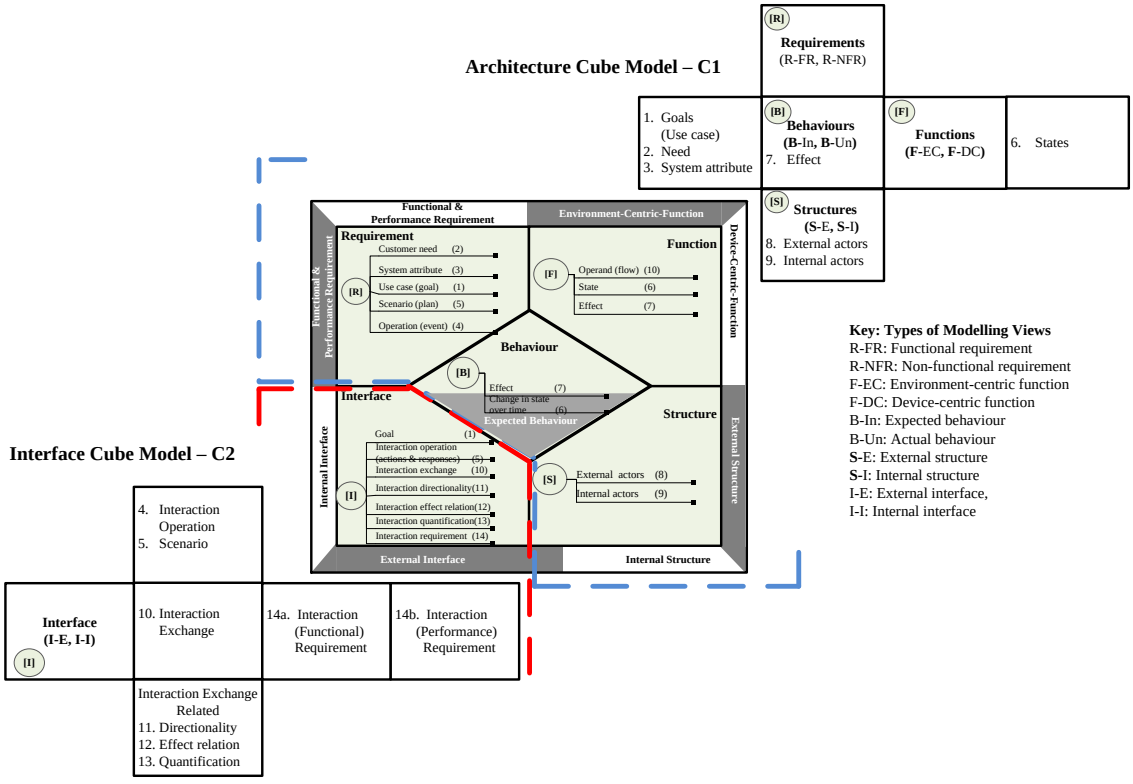


Figure 3.2 Two reference models derived from generic reference model

The C1-model is mainly a representation of architecture analysis model without interface view and comprising of other four views i.e. requirement [R], function [F], behaviour [B], and structure [S]. There are six common viewpoints as

discussed in Table 3.1. The two common viewpoints shared across pair of views (such as use case (in [R], [I]) and state (in [F],[B])) are placed separately with C1-model as shown in Figure 3.2. The other three common viewpoints i.e. exchange (as operand/flow), operation, and scenario already appear in C2-model. The effect viewpoint is kept with behaviour view in C1 model. The different types of each view are also specified in each view facet of C1 & C2 in the parenthesis in Figure 3.2 (such as F-EC & F-DC for environment-centric and device-centric functions) in order to assess whether any model or approach covers such types distinction.

3.3.2 From procedure perspective

In order to demonstrate the procedure of an approach, filled circles with numbers would be used with connected arrow headed lines. This would reveal the order of steps taken by an approach. It would also reveal which views or viewpoints bypassed in the procedure of an approach. Bold single headed arrows will also be used to show the route from starting to end point. The dashed arrow headed line would show which viewpoints of C2 model representable or equivalent in C1 model (or vice versa) and is thus perceived by an approach.

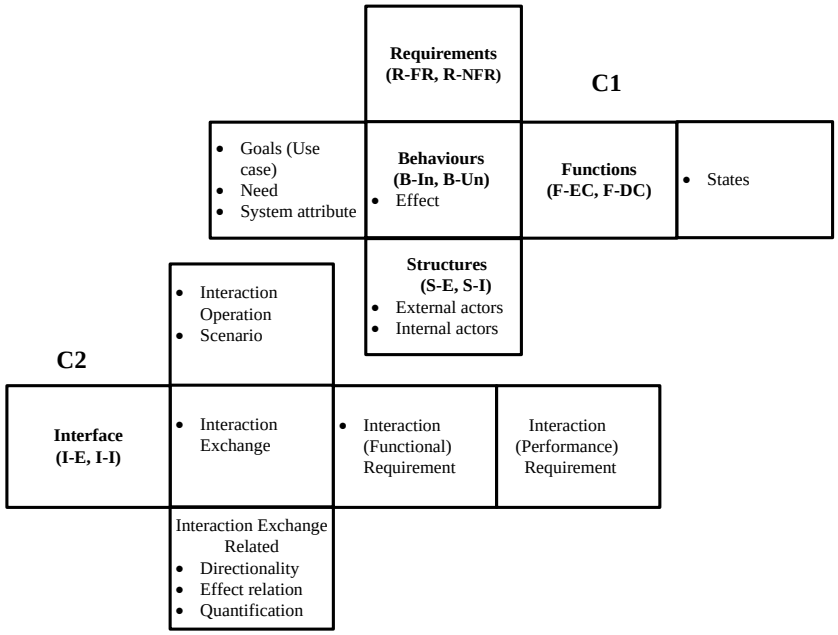


Figure 3.3 Visual criteria for evaluating existing approaches

Figure 3.3 (same as Figure 3.2 except numbers replaced by bullet points) will now be used as a set of criteria for evaluating each existing approach. Each

criterion is built on the characteristics (i.e. views, viewpoints and views types) of a technical system described in Chapter 2. The criteria and their rationales in the contexts of procedure and scope are as follow:

- *Does the approach represent the technical system view(s)? If yes, then what viewpoints are covered in it?*
- *Does the approach represent the types of technical system view(s)?*
- *Does the approach discuss its procedure iteration (at same hierarchical level) and recursive aspects (i.e. repetition to lower levels)?*

3.4 Evaluation of existing modelling approaches

The following section evaluates several well-known technical system modelling approaches as shown in Table 3.2 using the interface-cube and architecture-cube models (Figure 3.3).

Table 3.2 List of evaluated approaches from literature

S.No.	Evaluated approaches from literature
1	Quality function deployment
2	Buede's requirements-function coupling matrix
3	Bonnema's Funkey coupling matrix
4	Driessen et al.'s matrix based approach
5	Eisenbart's integrated function modelling approach
6	Vermaas's descriptive reasoning model
7	Daniels & Bahill's use-case modelling
8	Nilsen & Muller's use-case modelling
9	Eriksson et al's use-case modelling
10	Burge's context diagram model
11	Campean et al.'s Interface analysis table
12	Fosse & Delp's interface specification model
13	Fritzsche's interface description sheet

3.4.1 Evaluation of approaches supporting system architecture analysis

In this section, approaches that emphasise traceability and mapping between basic conceptualised views of a technical system are evaluated first.

3.4.1.1 Quality function deployment

Figure 3.4 shows the procedure and scope coverage of the quality function deployment for a technical system. The bold arrow headed lines show its procedural steps while dashed lines indicate a viewpoint from C2 equivalent to another viewpoint or coverage of specific type of a view in C1. The filled facets indicate scope coverage.

Quality function deployment is the comprehensive approach that maps two different viewpoints from one matrix to another. The ‘needs’ or ‘attributes’ of customer (as an external actor to a technical system) (step 1) are transformed into ‘engineering characteristics’ (step 2) (i.e. non-functional performance specification viewpoint) in the first matrix at system’s black-box without knowing its internals, as illustrated in Figure 3.4. After this first transformation phase, the ‘performance requirements’ (i.e. step 2) are allocated to system’s ‘internal components’ (i.e. system’s white-box) into the second matrix (i.e. step 3). This approach thus covers only ‘performance requirements’ (a type in requirements analysis). The customer language often seems to include a combination of needs and interaction operations (as highlighted in interface cube-C2 in Figure 3.4) of the user with the system. Furthermore, it captures very little information about interactions between system and its environment. Though the approach is quite robust, the definition (or distribution) of the ‘functions’ over the performance specifications (i.e. engineering characteristics) and components along with other viewpoints is not represented for system architecture analysis as shown graphically in Figure 3.4.

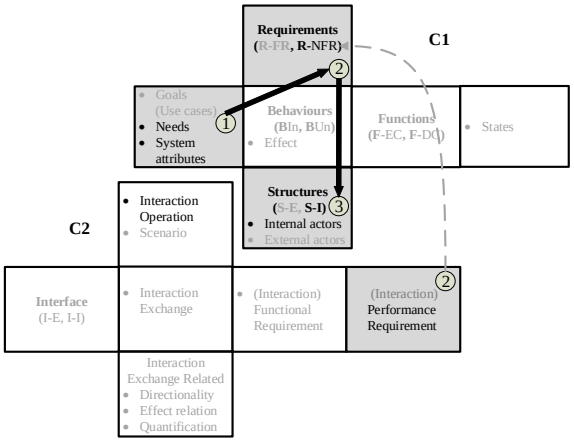


Figure 3.4 Scope and procedure of Quality function deployment

3.4.1.2 Buede's requirements-function coupling matrix

Buede (2009) presented a matrix-based approach that couples system's 'functions' to 'functional requirements' as illustrated graphically in Figure 3.5. The functional requirements on a system at black-box (including input/output and external interface requirements) are identified first (in step 1) and then are coupled with system's decomposed functions (in step 2). It does not provide information about 'non-functional (performance) requirements' in the same matrix in contrast to quality function deployment, as can be visualised in Figure 3.5. Also the matrix provides only objective domain thinking (i.e. technical; voice of engineer) from system perspective only and not the subjective thinking (i.e. voice of customer) from stakeholder perspective which is also essential as suggested in (Vermaas, 2009). Though other viewpoints are also discussed in (Buede, 2009) with different sets of tools, however a holistic integrated methodology is not provided.

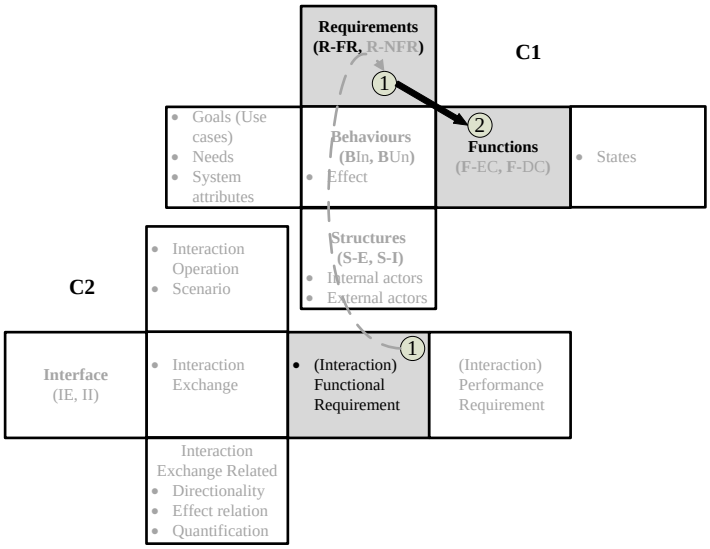


Figure 3.5 Scope and procedure of Buede's coupling matrix

3.4.1.3 Bonnema's Funkey coupling matrix

Bonnema (2008) provided a matrix-based FunKey architecting method where 'functions' that can contribute to stakeholders' 'key drivers/performance requirements' are mapped without the implementation in mind. It is then argued that the same set of functions and drivers/performance specifications can be achieved via multiple/alternative architectures comprising of external structure actors (i.e. system, user and environment at black-box) or allocated to a chosen

architecture's internals (i.e. subsystems at white-box) with budgets distribution as illustrated in Figure 3.6.

Although the FunKey matrix provides intuitive thinking, it does not discuss how stakeholders' drivers (and performance related aspects in there) are derived; however on one occasion, Bonnema (2011) stated that these can be seen as an *interface* between system developers (objective thinking) and stakeholders (subjective thinking). It further leads to an ambiguity in the sense that are these 'key drivers' representing 'stakeholder needs (i.e. key drivers)' or 'performance requirements' define by designers for a system and if both then it is not clear how this transition from drivers to performance aspects is achieved. Furthermore, the types related to function is not discussed as seen in the Buede's approach.

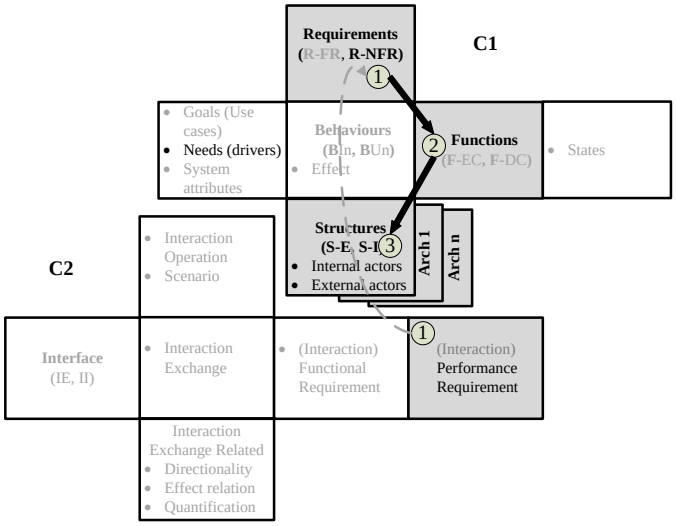


Figure 3.6 Scope and procedure of Bonnema's Funkey coupling matrix

3.4.1.4 Driessen et al.'s matrix based approach

Driessen et al. (2006), in Figure 3.7, show transition of key drivers to functional requirements and created a matrix based approach for software architecture's description in which 'key customer drivers' for the various markets are translated (coupled) into 'functional requirements'. The functional requirements are organised using a 'use case' viewpoint. Thus it can be concluded that use cases act as a binding element between customer drivers (needs) using natural language and system functional requirements. However, their matrix, does not

include coupling of ‘functional requirements’ to ‘functions’ (as seen in Figure 3.5) and to ‘performance requirements’ (as seen in Figure 3.6).

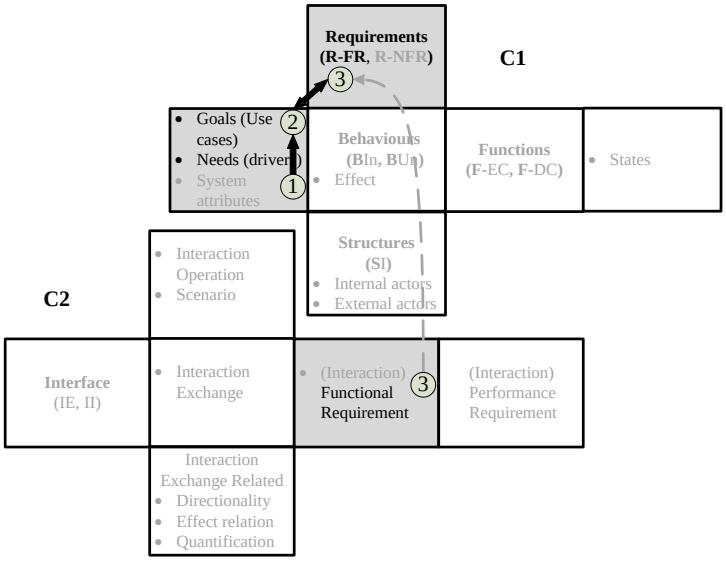


Figure 3.7 Scope and procedure of Driessen et al (2006) matrix approach

3.4.1.5 Eisenbart’s integrated function modelling approach

Eisenbart (2014) provides integrated function modelling approach where each matrix connects two different viewpoints but with a common view of ‘technical process’ (i.e. process definition similar to Hubka & Eder, 1996). This common view encapsulates two views i.e. behaviour and function as illustrated graphically in Figure 3.8. Also, process definition is categorised into two interaction (human processes) and transformation process (human processes, technical processes, and ‘effects’ related). According to Eisenbart (2014), modelling activities are carried out in the following procedural order: use case definition (step1), process flow modelling (step 2), operand state modelling (step 3), effect modelling (step 4), actor allocation (step 5), actor state modelling (step 6) and interaction specification (step 7) as illustrated graphically in Figure 3.8. The ‘process description’ in this approach seems to be closely related to ‘operation-based’ description of use cases.

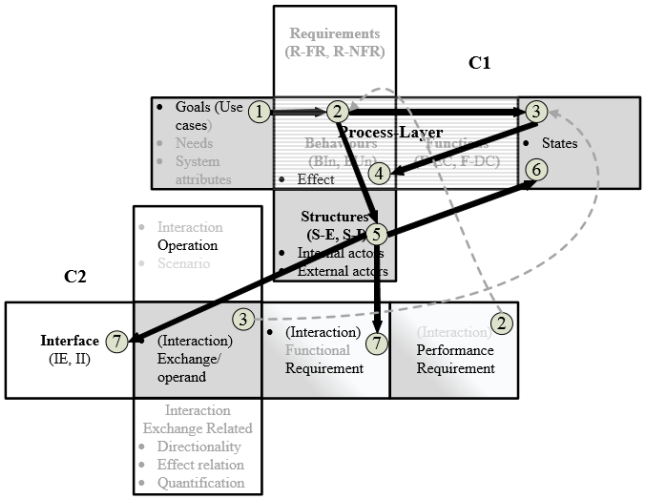


Figure 3.8 Scope and procedure of Eisenbart's integrated function modelling

The key problem with this approach is the definition and distinction of the interaction and transformation processes. When a 'process flow' of a system is created, it is not clear which ones are the *interaction* (i.e. input, and output specific related) and *transformation processes (transition from input to output related)?* How the distinctions in between these two are drawn in the '*process view*' at first place without defining both *operands* and external actors? though system is treated as a black-box. The distinction becomes clear later with operands states transition, and with internal and external *structure* actors mapping. Such aspect then violates the system-of-systems thinking i.e. one should start with external structure (black-box) around a system-of-interest and then exploring its internal structure that would satisfy its external structure. Moreover, in the state-matrix representation every process seems to be a transformative process. In that case, this approach then violates its own two basic process classifications. The specification of performance/constraint related aspects with process description is also emphasised as illustrated graphically in Figure 3.8 but how these are derived via what sort of path is also not clear.

It is worth noting that in engineering design community the operands are identified first (as fundamental recommended approach by well-known theories) and based on their states transition thinking, the top and sub-functions in (Pahl et al, 2007) or processes in (Hubka & Eder, 1996) are identified i.e. looking at operands specifications. In the Eisenbart approach, contrary to these basic

fundamental recommended approaches, the processes are identified first and then those are arranged later into states transition as evident from Figure 3.8.

3.4.1.6 Vermaas's descriptive reasoning model

Vermaas (2009; 2010) presented mainly conceptual views/concepts for the engineering descriptions of a technical system. He also suggested two possible paths; top-down and bottom-up in these five concepts. The top-down path is illustrated in Figure 3.9. For the bottom-up, the path should be other way round as per Vermaas' arguments.

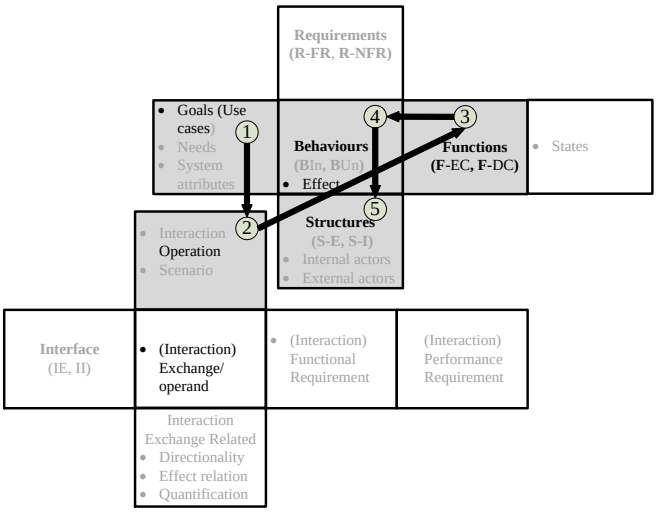


Figure 3.9 Scope and procedure of Vermaas's reasoning model

Brown & Blessing (2005) reasoning model also fits till step 4 in Figure 3.9. However, in these models, interface view and requirement view's related types are less accentuated.

3.4.1.7 Summary

It seems that the existing system architecting approaches are insufficient in scope to model the technical system's views [R], [F], [B] and [S] based on the criteria in architecture-cube model C1.

Note that the existing approaches, evaluated so far, support system architecture analysis but discuss very little on the role of interface modelling view of a technical system after looking at interface-cube model C2 with model C1. Interfaces play a vital role in system architecture analysis. The only integrated function modelling framework (Figure 3.8) discuss interfaces and interactions

but at system's white-box view when both external and internal actors are known. The approaches that deal with requirements derivation via system interfaces and interactions modelling at black-box and white-box views are also evaluated in next section.

3.4.2 Evaluation of approaches supporting interfaces and requirements analysis

3.4.2.1 Approaches - modelling interactions as operations

In this section, the existing approaches are evaluated that emphasise the derivation of system requirements via sequence of operational events for achieving its use cases.

3.4.2.1.1 Modelling interactions as sequence of operations at black-box

3.4.2.1.1.1 Daniels & Bahill's use-case modelling

The order of procedural steps of Daniels & Bahill (2004) approach is shown in Figure 3.10. In step 1, external actors are identified and linked to use cases in step 2. The double arrow head indicates that external actors to system are linked to relevant use cases (and thus the steps 1 & 2 can be other way round). The system's 'shall' based requirements (both functional and non-functional) are derived (step 4 & 5) from interaction events (step 3) occurring between system (i.e. black-box) and its external actors in main success (sunny) or alternate flow (rainy) scenarios to achieve a use case as illustrated in Figure 3.10. A main success or sunny scenario describes normal way of achieving a goal whilst alternative flow scenario shows alternate ways of achieving a goal or any failures handled by system. These sequence of events are often described as 'system behaviour' by Daniels & Bahill (2004). They use use-case diagram and textual template for this purpose.

Daniels & Bahill do not show how this approach can be recursively applied to lower levels or at system's white-box. Daniels & Bahill also discuss very little on distinct relationship between the two types of derived system requirements.

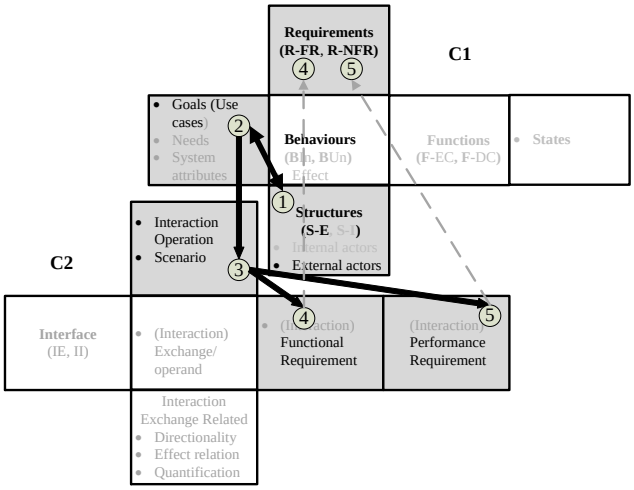


Figure 3.10 Scope and procedure of Daniels & Bahill's approach

3.4.2.1.1.2 Nilsen & Muller's use-case modelling

Nilsen & Muller (2014) combine non-functional performance requirements and use cases (to them functional requirements) in a use case diagram. The external actors are linked to use cases (step 1) as illustrated in Figure 3.11. They show that a single use case as functional requirement (step 2) can have multiple non-functional performance requirements (step 3). However, on the other hand, the approach lacks to show explicit linkage and discussion with scenarios and sequence of events explicitly as illustrated graphically in Figure 3.11 in contrast to Daniels & Bahill's use case model approach.

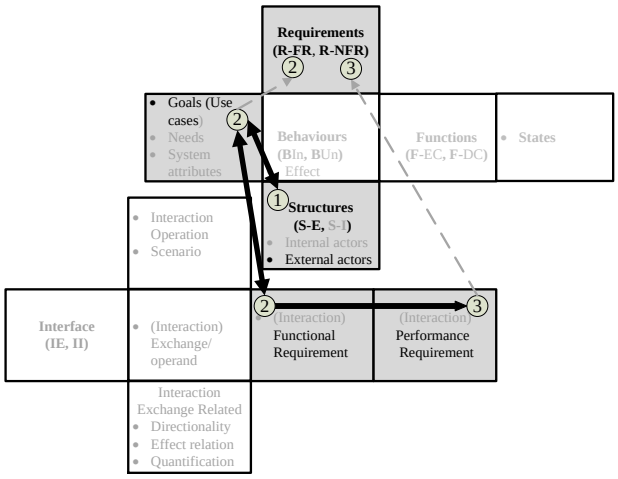


Figure 3.11 Scope and procedure of Nilsen & Muller's approach

3.4.2.1.2 Modelling interactions as sequence of operations at white-box

3.4.2.1.2.1 Eriksson et al.'s use case modelling

Eriksson et al (2008) approach begins with external actors' interfaces identification (i.e. step 1) using context diagram, their linkage to use cases via use case diagram (step 2), and then defining sequence of operations as actions and responses at system's black-box with budgeted requirement field in RUP template (step 3 & 4) as illustrated in Figure 3.12.

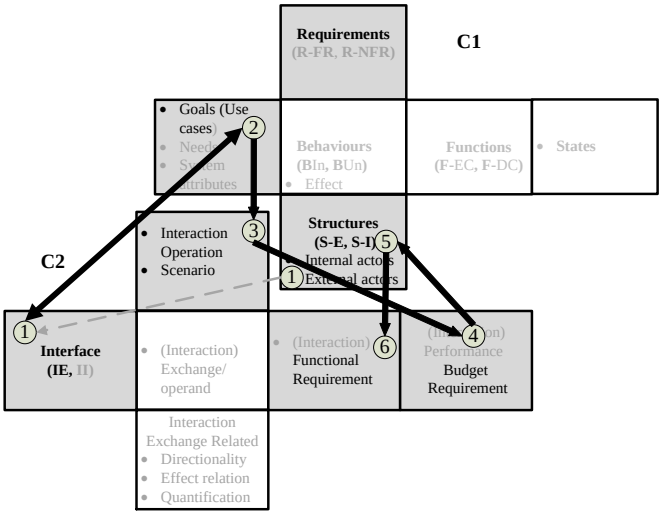


Figure 3.12 Scope and procedure of Eriksson et al.'s approach

The black-box system response events are then broken down to white-box actions of its internal subsystems (step 5) and from there, subsystems requirements are derived (step 6). Furthermore, budged requirement field (analogous to performance related) helps in deriving robustly 'shall-based' functional requirements for subsystems (i.e. at system's white-box). However, their budgeted requirement is just restricted to time parameter while there can be many other performance requirements as discussed by Nilsen & Muller (2014). Furthermore, Eriksson et al. do not distinguish as such between functional and (non-functional) performance requirements types as illustrated graphically in Figure 3.12.

3.4.2.2 Existing approaches - modelling interactions as exchanges

3.4.2.2.1 Modelling interactions as exchanges only at black-box

3.4.2.2.1.1 Burge's system context diagram model

Burge (2011) and Kossiakoff et al (2011) also recommend of identifying interactions as exchange/flow viewpoint (step 3) at system interfaces (step 2) with its surrounding actors identification (step 1) via a context diagram. The high level operational functional description at black-box is also recommended as illustrated in Figure 3.13. Similar strategy is applied at black-box level by engineering design community researchers such as Pahl et al (2007), and Otto and Wood (2001). However, in all these, no discussion is observed on the sequence of identifying exchanges at black-box view.

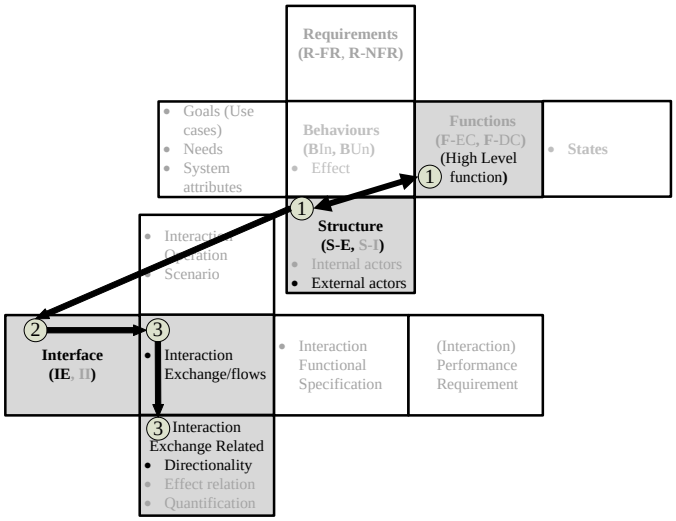


Figure 3.13 Scope and procedure of Context diagram

3.4.2.2.2 Modelling interactions as exchanges only at white-box

3.4.2.2.2.1 Campean et al.'s interface analysis table

Campean et al. (2011) aim to derive functional requirements at system's white-box based on identified energy, material, information and physical exchanges between two known internal interacting actors and between internal and external actors' interfaces thereby considering exchanges impact on system main functions. However 'performance specifications' viewpoint and 'interaction operations' are not discussed explicitly as shown in interface-cube C2-model in Figure 3.14. The function viewpoint is associated with architecture-cube C1-model.

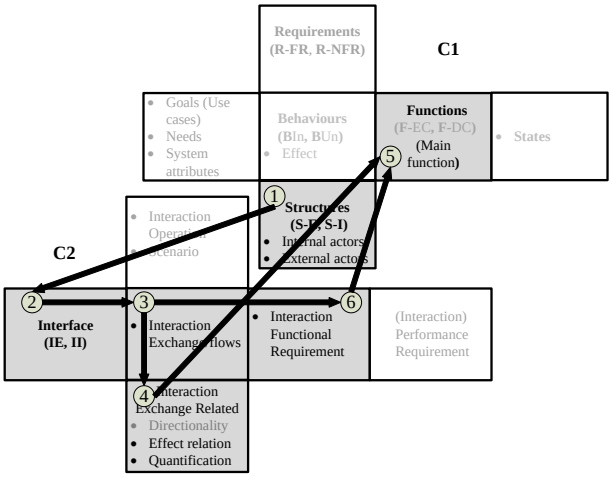


Figure 3.14 Scope and procedure of Campean et al.'s approach

3.4.2.2.2.2 Fosse & Delp's interface specification model

Fosse & Delp (2002), in Figure 3.15, use SysML to describe the interface requirements robustly, however impact on main system functionality is not shown in contrast to Figure 3.14. Campean et al (2011) and Fosse & Delp (2002) do not recommend identifying exchanges in a particular sequence.

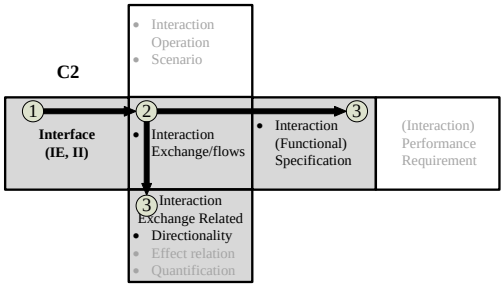


Figure 3.15 Scope and procedure of Fosse & Delp's approach

3.4.2.2.2.3 Fritzsche's interface description sheet

Fritzsche (2008) shows interface description sheet that define interface requirements between system's subsystems and external actors at system's white-box view. Performance specification is specified first based on exchange description and then functional requirement with it, as illustrated in Figure 3.16. However, this sheet does not show linkage to high level main function(s) as observed in Figure 3.14.

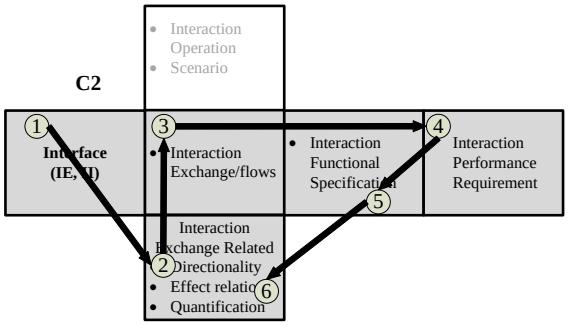


Figure 3.16 Scope and procedure of Fritzsche's approach

3.4.2.3 Summary

It is seen and evaluated that each of existing system modelling approaches seem to be incomplete and inconsistent from *scope perspective* in the context of requirements [R] and interface [I] views. The examined approaches from Figure 3.10 to Figure 3.16 seem to be suitable for interactions modelling; however these emphasise less on linkage with other modelling views of an architecture analysis as well as their applicability across multiple hierarchical levels.

It is also seen in existing approaches that it is hard to comment on the correctness of the *procedure* due to different entry points though similar viewpoints are used and adopted. Overall, none of the existing approaches completely cover the conceptualised characteristics of a technical system or suggest consistent and correct procedural guidelines in detail.

3.5 Research gaps

The gaps in the existing state of art are summarised in this section. The research hypothesis are then established.

3.5.1 Critique of approaches in the context of system architecture analysis

Following are the critical gaps seen in existing approaches looking at Figures 3.4 to 3.9.

- ***Lack of structured integrated interface modelling view***: It becomes quite evident from Figure 3.4 to 3.9 that the role of interface view and the relevant interface modelling viewpoints are not emphasised or integrated

in a structured manner with other system architecture analysis views by the existing approaches. McDavid (2005) stated, an architectural approach should emphasize the need for multiple levels of abstraction with interfaces clearly defined in a structured manner.

- ***Incomplete mapping or relationships between viewpoints****:* It becomes apparent from Figure 3.4 to Figure 3.9 that a clear relationships needs to be defined between 'drivers (as needs)', 'use cases', 'functional requirement', 'performance requirements' and 'functions'. The matrix based approaches from Figures 3.4 to 3.9 often bypass one of these four key viewpoints. The existing approaches also suggest that there can be either one-to-one or many-to-many relationships between the mapped views or/and viewpoints.
- ***Partial views are represented by existing approaches:*** Many of the reviewed approaches aim to cover minimum two and maximum three system architecture views in a combined resulting matrix. For example, [R] and [S] views by quality function deployment whereas [R], [F], and [S] views by Funkey coupling matrix.
- ***Incomplete traceability between levels:*** The iteration and traceability between modelling views is emphasized at one level; however traceability across multiple levels (i.e. recursive aspect) is not discussed by most of the approaches except Funkey coupling matrix.
- ***Insufficient coverage of views' types:*** It is also emphasized by existing approaches that functions should be defined independent of any specific solution that could support multiple architectures (Figure 3.6). However, the reasoning on functions at first place is not discussed in such approaches as if a function is environment centric? Or device centric?
- ***Insufficient guidelines for views development:*** It is also observed that models like Vermaas provide system's functional reasoning paths in a top-down and bottom-up ways but does not suggest any structured holistic methodology among the conceptualised views of a technical system. Such reasoning models describe '*what'* needs to be done or has done in what sequence among the basic conceptualised views in literature. There is quite a lack of guidelines or instructions of '*how'* to do it.

3.5.2 Critique of approaches in the context of system interface and requirements analysis

- ***Incomplete procedure of requirements analysis***: The evaluated approaches from Figure 3.10 to Figure 3.16 help in deriving functional and performance requirements clearly in contrast to the approaches evaluated from Figure 3.4 to Figure 3.9. However, few approaches show one-to-many distinct relationships between functional and performance requirements (Figure 3.11) but do not discuss the derivation of these two. On the other hand, few approaches show derivation of these two types (Figure 3.10) and not a distinct relationship in between the two types of requirements.
- ***Inconsistency in templates and partial viewpoints consideration for capturing system interaction requirements:*** On one hand, many approaches emphasize identification of multiple types of ***interaction exchanges*** at system's black-box (at requirements analysis stage) with its external actors (Figure 3.13) whilst others capture them at white-box (at physical architecture stage) (Figures 3.14 and 3.16). However, none of these suggest any particular order or sequence of identifying exchanges both at system's black and white-box views. The templates that capture interface requirements at system and subsystem levels vary due to consideration of different viewpoints (Figures 3.13 to 3.16). It means there is no standard content and concrete form for interface analysis.
- On the other hand, some approaches emphasize identification of sequence of events as ***interaction operations*** between system (by treating it as a black-box) and its surrounding actors via use case-scenario templates and use them for deriving system functional and performance requirements (e.g. Figure 3.10). Similar logic is applied at system's white-box view to derive subsystem requirements (e.g. Figure 3.12). However, there is a diversity in the templates and viewpoints for use case modelling (from Figures 3.10 to 3.12). This leads to the fact that there is no standard template in use case modelling for capturing system requirements. The above two bullet points are summarised in following Table 3.3.

Table 3.3 Summary of interaction modelling viewpoints

Interaction viewpoints across both views	System conceptualised views	System modelling viewpoints	System views on a design hierarchy
As Events (i.e. Operations) Or/and As Exchanges across both levels	Functionality related	• Functional Requirements & • Performance Requirements	Black Box view
	External structure	• System & its interactions with the External Actors	
	Functionality related	Solutions Dependent • Functional Requirements • Performance Requirements	White Box view
	Internal structure	• Internal Actors & their interactions with External Actors	

Table 3.3 summarises that system's black-box view (i.e. its external structure) is analysed either via interaction operations or via exchanges by existing approaches. Based on these two separate viewpoints, system requirements (both functional and non-functional) are derived at black-box. Similarly, same two viewpoints are utilised at system's white-box (i.e. its internal structure) to derive functional and performance requirements. Though, both these viewpoints i.e. interactions as flows/exchanges (often use by mechanical engineers) and interactions as operations (by software engineers) are equally important at system's interfaces, none of the existing approaches show a distinct relationship in between these two viewpoints both at black-box and white-box views.

- ***Incomplete consideration of design thinking perspectives:*** It is also observed that two different mind-sets thinking is also promoted in interface analysis sheets in order to differentiate the stakeholder's subjective thinking (i.e. voice-of-customer) from technical system's (i.e. voice-of-engineer or designer) objective thinking. Such thinking is not promoted in existing approaches step-by-step.
- ***Incomplete guidelines for iterative and recursive aspects:*** Furthermore, concrete guidelines for the usage of available templates for requirements and interface definition are not provided in the context of iterative and recursive aspects except RUP template (Figure 3.12).
- ***Consideration of stakeholder related system attributes:*** None of the evaluated approaches show grouping of requirements according to

system attributes related to stakeholders or assess the effect of interactions on the system attributes except quality function deployment.

- ***Lack of common interface documentation language support***: ICDs make projects document driven with contents and working style varying from one disciplinary team to another, that causes difficulty in collaboration. This leads to the fact that there is no common unanimous agreed contents and template at first place that could encapsulate the language of each discipline (e.g. see contents and working style diversity from Figures 3.10 to 3.16).

3.6 Research directions

3.6.1 Research contribution in interface modelling

The findings and the discussed problems with the existent interface and requirements modelling approaches reveal that there is no shared interface definition model. Based on identified gaps in interface modelling approaches, the following research hypothesis is established:

Research hypothesis 1: *Modelling system interactions with diverse viewpoints at its interfaces will lead to an* ***effective and complete*** *approach for defining* ***system interfaces*** *and deriving* ***requirements.***

Figure 3.17 Research hypothesis 1

3.6.1.1 Requirements on interface modelling approach

The interface analysis viewpoints, and the following requirements are the key driving factors for the development of a novel interface modelling approach.

1. ***Different viewpoints integration:*** the approach should incorporate and integrate diverse but essential viewpoints of interface analysis with distinct relationships among those viewpoints (e.g. one-to-one or one-to-many) that are often not considered in a single template or approach.
2. ***Multidisciplinary and system architecture analysis support:*** the approach should be implementable across multidisciplinary environment, and should support all sorts of system's interfaces (such as electrical, mechanical, and control related) within any design hierarchy, and also should support other system architecture views.

3. ***Abstract to detail view support:*** the approach should provide a flexibility and support to define an interface in increasing order of detail i.e. from an abstract information to detail information, with or without skipping any viewpoint.
4. ***Multiple stakeholders' language support:*** the approach should provide a common language support to software engineers (using use case analysis) and electromechanical engineers (using exchange-based analysis), to document the requirements, working together for a system underdevelopment from different disciplines thus reducing the risk of communication errors or ambiguous information.
5. ***Documentation support:*** the approach should improve documentation process and allow on paper to switch between and update diverse viewpoints quickly rather than following a software based tool (such as UML/SysML) that may require analysing a system with different viewpoints where each viewpoint requires development of a separate diagram.
6. ***Application versatility and flexibility:*** the approach should be flexible enough and allow freedom to multidisciplinary engineers to adopt their own style of identifying system interactions thereby leading to a similar set of outcomes i.e. identifying robust set of requirements on their systems with robust formalism in procedure.

3.6.2 Research expansion in system architecting approach

The findings and the described problems with the existent system architecting approaches also reveal that there is no shared system architecture analysis model in conjunction with interface modelling approach. Mostly approaches cover system architecture views from requirements definition and their allocation to function till structure views and discuss very little on integration of interface view aspect in detail. Based on this key gap in relation to architecting approaches, the following research hypothesis is established:

Research Hypothesis 2: *Modelling **system architecture in conjunction with well-defined interfaces** will lead to an **effective** architecting approach for requirements allocation and information traceability across system's multiple views.*

Figure 3.18 Research hypothesis 2

It is also envisioned that integrated interface modelling approach with system architecture analysis approach should support iteration and recursive aspect in system-of-systems thinking i.e. across multiple levels of abstraction/decomposition. Therefore, following research hypothesis is also established:

Research Hypothesis 3: *An architecting approach in conjunction with well-defined interfaces would support **consistent** system-of-systems thinking i.e. **multiple levels of abstraction** (decomposition) analysis.*

Figure 3.19 Research hypothesis 3

3.6.2.1 Requirements on system architecture analysis approach

The system architecture analysis viewpoints (including interface view), and the following requirements are the key driving factors for the development of a novel architecting approach.

1. ***Different modelling views integration support:*** the approach should incorporate necessary system architecture views and integrate essential viewpoints with structured relationships (e.g. one-to-one or one-to-many) that are often not considered as a part of one combined framework;
    - Allow for modelling between system functions and interaction functional and performance requirements;
    - Allow for modelling between system functions and design subsystems.
2. ***Multidisciplinary and system architecting support:*** the approach should be implementable across multidisciplinary environment, and its working should remain same across various levels of design hierarchy
3. ***Modularity support:*** the approach should support all types of functional modelling tools such as function-flow diagram, mind-mapping diagram,

and state-transition representation diagram i.e. it should be compatible with any functional modelling tool.

4. ***Modular & integrated architecture:*** the approach should support both modular and integrated architecture types. In the modular architecture type, a system's subsystems/modules are functionally self-contained (i.e. one-to-one mapping e.g. one function served by one subsystem), whereas in an integral architecture type subsystems are functionally tightly coupled (i.e. one-to-many mapping e.g. many functions served by one subsystem and alternately one function by many subsystems) (Ulrich, 1995).
5. ***Traceability:*** the approach should support traceability between different views or viewpoints on the same page at one decomposition level among following:
    - Allow for mapping between use cases, functions, interaction functional and performance requirement, and subsystems.
6. ***Reusability and Upgradeability:*** the approach should be flexible enough to update (or delete) or add any additional information for new solutions by utilizing the information available for previous solutions on paper.

The next section describes the research methodology to draw out and test the both theoretical and empirical consequences of the research hypothesis.

3.7 Research methodology

The theory and empirical research aspects are discussed by Grix (2004) and Davis (2006). As outlined by them, theory-driven research is often referred as deductive research which is based on research hypothesis that are tested on empirical data. Empirical-driven research involves empirical data on which research is refined and conclusions are drawn. Figure 3.20 shows the steps of research methodology used in this research.

		Case Studies Features					Thesis Chapters						
		Multidisciplinary Applications			Application Levels								
		Software Application	Mechanical Application	Electro-Mechanical	One level of decomposition/ abstraction	Multiple levels of decomposition /abstraction	Chapter 1	Chapter 2	Chapter 3	Chapter 4	Chapter 5	Chapter 6	Chapter 7
Research Methodology Roadmap													
1. Problem Formulation							X						
2. Literature Review								X					
3. Research Gaps & Hypothesis									X				
3.1 Research Hypothesis									X	1	2	3	
4. Research Case studies Execution										X	X	X	
4.1 Desktop Cases	Ball Pen System		X		X					X	X		
	Electric Sharpener			X	X					X			
	Coffee Vending Machine			X	X						X		
4.2 Real Cases	Surround View Feature	X								X			
	Regenerative Braking System			X		X						X	
5. Results Analysis & Hypothesis Testing												X	
5.1 Effectiveness evaluation												X	
5.2 Usefulness across academia and industry													X

Figure 3.20 Roadmap of research methodology

(1) Formulating the research problem*:* The research began by 'formulating the research problem' around the research fields of interface, architecture analysis, and multiple levels of decomposition/abstraction (i.e. system-of-systems thinking).

(2) Literature review*:* The literature was reviewed in the selected fields thereby considering relevant approaches and aggregating relevant information from published academic journals, conference proceedings, and books mainly. The key focus was on understanding the role of interfaces in the system architecture analysis across multiple levels of abstraction for which thorough literature review conducted on system architecture and its relevant modelling views, viewpoints, and conceptual black-box and white-box views.

(3) Research gaps and development of hypothesis**:** Based on literature review, it was observed that there is no concrete reference model for system architecture analysis (encapsulated with modelling views, viewpoints, and box views at one level of decomposition/abstraction) that could be used for evaluating existing approaches in their scope and procedure. This resulted in the development of the reference model in this research first. The interface-cube and architecture-cube models were then derived from the reference model to assess the existing approaches on equal merit in their fields. These models helped in evaluating existing approaches and identifying gaps in the interface

and architecture analysis fields and concluded mainly that integration in these two fields has not yet been fully accomplished. Based on the identified gaps, three research hypothesis were established (as presented in previous section).

(4) Research design execution- case studies driven: To address the research hypothesis in maximal information, this research adopted a range of case studies ranging from simple (desktop based) to complex (real world based) as well as monodisciplinary to multidisciplinary applications for developments and validation purposes. The desktop case studies were initially used for development purposes. Later, real world complex case studies were conducted with an automotive company for the purpose of empirical validation and gaining practical results.

(4.1) Desktop driven research: The interface-cube model-C2 set the ground for a theoretical development of interface modelling methodology. The concepts of developed methodology were then tested with a range of desktop applications. The simple ball point pen case study (purely monodisciplinary i.e. mechanical application) was used as an application for reasoning purpose with an advantage that this case study has been analysed in literature with mostly viewpoints & concepts accumulated in the cubes-models C1-C2 (Figure 3.3). The reasoning of each viewpoint with each step of interface methodology yielded to a novel interface definition tool named as Interface Analysis Template (IAT). The IAT was then validated with relatively complex desktop example (in contrast to ball point pen) i.e. Electric Pencil Sharpener (a multidisciplinary application i.e. electromechanical application). In this thesis, the first hypothesis is addressed in Chapter 4 as shown in Figure 3.20.

The architecture-cube C1 model (Figure 3.3) is mainly used for theoretical development of architecture analysis framework in conjunction with IAT tool named as Coupling Matrix (CM). The initial reasoning for CM framework is done via same ball point pen example followed by its validation on a relatively complex desktop case study: Coffee Vending Machine (an electro-mechanical application). The second research hypothesis of this research is described in Chapter 5.

At this stage of research, the developed IAT tool and CM framework were validated mainly on desktop case studies but only at one level of decomposition/abstraction i.e. from level-0 to level-1 (Figure 2.7).

***(4.2) Real world driven research*:** The integrated architecture analysis framework was then proposed encapsulating both IAT tool and CM framework which was then validated with an automotive company on three different complex automotive applications that involved a set of seven independent engineers split in three teams across different levels such as feature level, system level, and subsystem level in a vehicle hierarchy.

The IAT was validated with Team 1 dealing with Surround View Feature (i.e. software application) at feature level, and also with Team 2 that dealt with Deployable Active Rear Spoiler System (i.e. electro-mechanical application) at subsystem level of a vehicle. The IAT validation with these case studies was limited to one level of decomposition/abstraction i.e. from level-0 (black-box) to level-1 (i.e. white-box of level 0) (Figure 2.7). The CM was also validated with Team 2's project i.e. Deployable Active Rear Spoiler system.

The Team 2's project i.e. Regenerative Braking System (an electromechanical application) at system level was used to validate the integrated architecture analysis framework based on IAT and CM across the multiple levels of abstraction (i.e. from level-0 to level-1 to level-2 – Figure 2.7). The third research hypothesis was related to this application and is addressed in Chapter 6.

***(5) Research effectiveness evaluation**:* The empirical study setup helped in analysing the practicalities and gaining useful results on the effectiveness of IAT tool and CM framework. The results were also related to three research hypothesis and are presented in Chapter 6. The usefulness of this research in the context of academia and industrial sectors is then discussed in Chapter 7.

3.8 Chapter summary

This chapter has developed a system architecture reference model which is used to outline the limitation in existing approaches both from scope and procedure perspectives in the fields of architecture and interfaces analysis. The research gaps, hypothesis, and the requirements for the development of

needed tools/methods are also highlighted. The research methodology is also introduced for testing the research hypothesis.

Chapter 4 introduces the interface analysis template as a novel interface modelling methodology to support system architecture analysis thereby meeting the requirements of Section 3.6.1.1. Chapter 5 presents a coupling matrix in conjunction with interface analysis template to support system architecture analysis thereby meeting requirements of Section 3.6.2.1.

4. Development of an Interface Modelling Methodology

4.1 Introduction

The aim of this chapter is to introduce an interface modelling methodology based on diverse viewpoints integrated in interface-cube model (Figure 3.3) which has revealed that existing approaches are limited in scope and procedure (Section 3.4.2). A literature review example will be used to justify the proposed methodology with different viewpoints analysis step-by-step thereby discussing distinct relationships in between them at interfaces with strong reasoning.

Based on reasoning in the first half, a novel tool named as an interface analysis template (IAT) is developed and tested via a desktop case study i.e. Electric Pencil Sharpener. It is then shown how IAT can be applied consistently with same template across the system's external (black-box view) and internal interfaces (white-box view). The guidelines for using the IAT in different design scenarios are also presented. In the end, the strengths and weaknesses of IAT are highlighted based on validation results obtained from automotive industry.

4.2 Development of a methodology for interface definition

A structured interface definition methodology is developed and presented in this section in order to meet the requirements highlighted in Section 3.6.1.1.

4.2.1 The interface modelling viewpoints and methodology steps

The different viewpoints, in Figure 4.1a, that are used across interface and requirements views (Table 3.1), arranged into six-step based interface definition methodology as shown in Figure 4.1b. In the next section, each step of the methodology along with two loops in it are discussed.

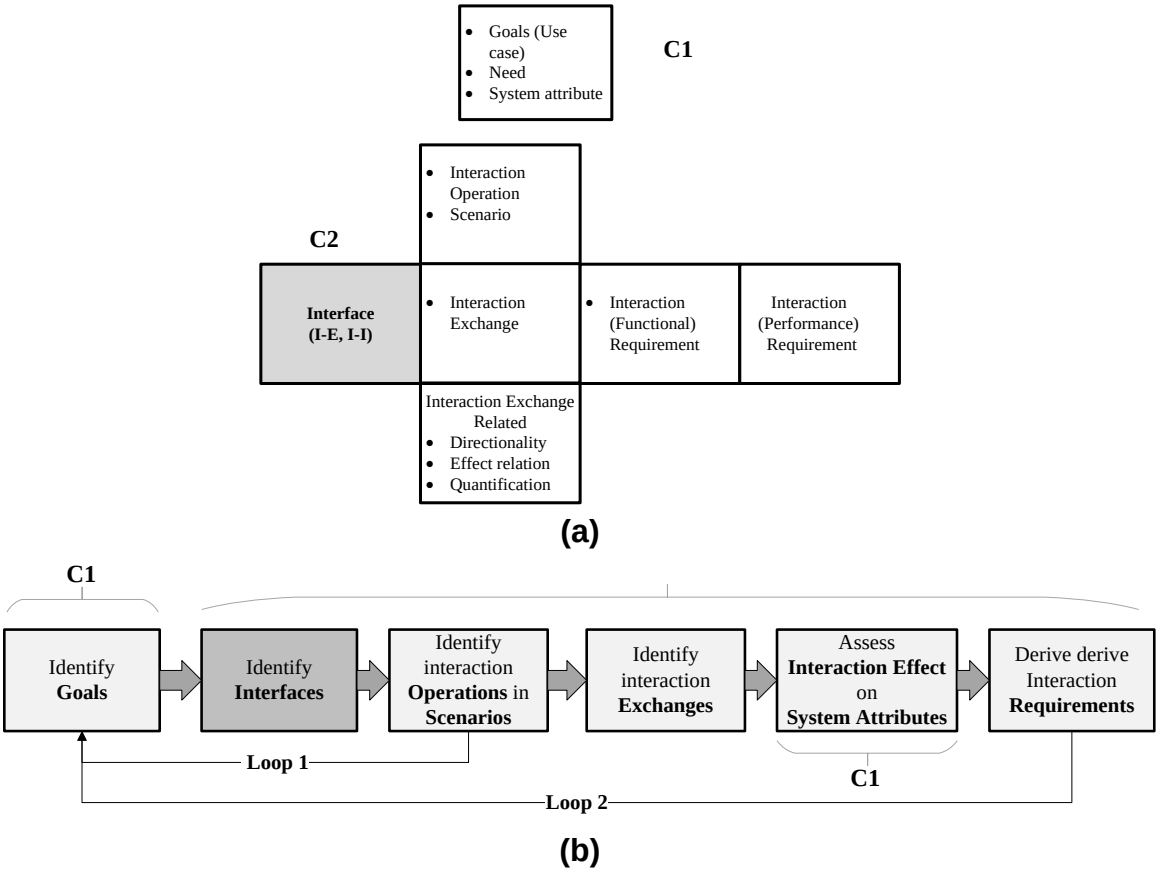


Figure 4.1 Proposed six-step based interface modelling methodology

4.2.2 Reasoning on proposed methodology

4.2.2.1 Introduction to pen device: An exemplary system

In this section, the methodological steps are introduced and discussed with a well-known example 'ball-point pen' presented by several researchers (Crilly, 2012; Brown & Blessing, 2005; Vermaas, 2009). It is discussed that the same interface modelling viewpoints have been perceived by different researchers in a slightly different manner.

The existing data on ball point pen example, in Figure 4.2, is taken from Brown & Blessing (2005) to discuss a set of mutually exclusive and collective exhaustive viewpoints (Figure 4.1a). The key reason behind the selection of this example is the availability of analysis and definition of modelling viewpoints by other researchers so that it provides a foundation for strong reasoning for the development of novel IAT tool.

D: Pen.

Structural element: tip.
Structural element: ink container.

Structural connection: tip is at the end of the ink container.
Structural connection: tip is connected to the ink container.

State variable: pressure.
State variable: orientation.
State variable: location.

Mode of Deployment: human is gripping pen; pen is tip down; tip is in contact with paper; the tip exerts pressure on the paper.

B: ink flows from the tip; ink coats the paper; the tip is moving.

Goal: to have another human know the information that you want to tell them.
Intention: get paper, get pen, write message, transfer paper to other human.
Plan: grip pen, orient pen, put pen tip to paper, apply pressure, move pen.

Device-centric function: The function of the pen is to cause ink to flow out of its ink container onto the tip.

Mixed function: The function of the pen is to cause ink to flow from the ink container to the tip, and onto some paper.

Environment-centric function: to cause a piece of paper to have ink on it.
Environment-centric function: to write.
Environment-centric function: to communicate information.

Figure 4.2 Pen example (adopted from Brown & Blessing, 2005)

As can be seen from Figure 4.2, the different conceptualised viewpoints on pen device by Brown & Blessing (2005) are covered in sufficient detail but they do not discuss a formalised structure of a holistic methodology or tool that could organise or reveal the order of such viewpoints in a correct and complete manner step by step. Therefore, in the next section, reasoning on interaction modelling viewpoints is discussed to support the derivation of IAT tool.

4.2.2.2 System engineer's holistic thinking perspectives

The system engineer (or designer) often goes through two different mind-sets for the analysis of a system: stakeholders' perspective (i.e. voice-of-customer) and system-of-interest's perspective (i.e. voice-of-engineer) as discussed in Chapters 2 and 3, and as illustrated graphically in Figure 4.3. These perspectives are often referred to as subjective (stakeholder domain) and objective (technical domain) in (Vermaas, 2009) where engineer switches from thinking in one domain to another.

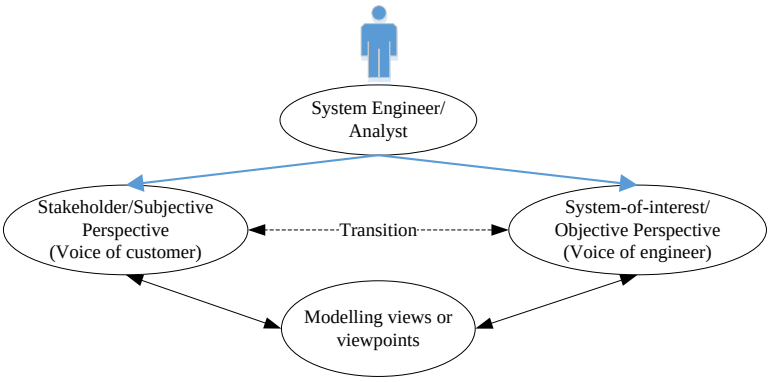


Figure 4.3 System engineer analytical thinking perspectives

4.2.2.3 Step 1: Identify goals

The customer needs are considered as a starting point for engineering design activities. These are often considered hard to interpret from a technical perspective. Therefore researchers often consider use cases (Figure 4.1a) that describe the goals of the system with respect to its stakeholders (end customer and other parties), thus use cases allow the system engineer to think of or link system goals from stakeholders' or its external environment perspective. Note the use cases were discussed as a bridge between voice of stakeholder and voice of engineer (Section 2.4.3.2.1).

In order to design and analyse a system-of-interest, it is recommended in literature to identify the system context in relation to its goals with stakeholders. In this research, system design analysis begins with identification of goals and relevant stakeholders or external actors. The external actors reveal the *external structure* of a system-of-interest.

Brown & Blessing (2005) and Vermaas (2009) describe the following goal for the normal ball point pen device usage in social society;

> *Device Goal 1 (G1): "to provide information to another person" (Figure 4.2).*

A goal is related to its relevant actor. In this case, the goal – G1 is potentially associated to external actors i.e. user, reader and system-of-interest being pen device.
If the designer extends the scope by bringing-in another external actor in the defined context of external structure of a system, there can be many other goals to include. For example, thinking of utilising a pen device for other following purposes:

> *Device Goal 2 (G2): to advertise a business [associated with external actor – the institute/company, user, and the pen],*
>
> *Device Goal 3 (G3): to carry through [associated with transporter, user, and the pen].*

Therefore, the goals can also vary with the introduction of a new external actor as discussed in above two examples. This section has discussed the 'goal' viewpoint (Figure 4.1a).

4.2.2.4 Step 2: Identify interfaces

Once the goals and relevant associated actors are identified then one can state that system external interfaces with its environment are identified. Thus, goals and relevant external actors can be listed in a tabular format as shown in Table 4.1 and is summarised graphically in Figure 4.4.

Table 4.1: Goals and interfaces relationships of a pen device

Step 2: Device interfaces	**Step 1: Device goals**		
Mode of Deployment M (D,E)	**G1: To provide information**	**G2: To advertise a business**	**G3: To carry through**
M (Pen, User)	X	X	X
M (Pen, Reader)	X		
M (Pen, Paper)	X		
M (Pen, Company)		X	
M (Pen, Transporter)			X

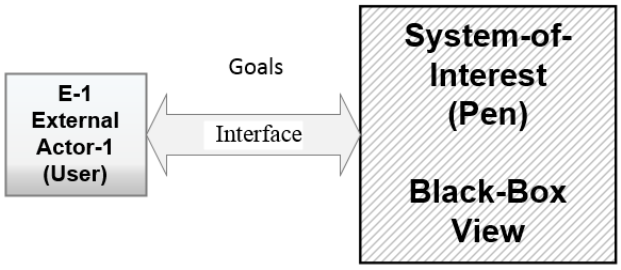


Figure 4.4 System-of-interest's goals and interfaces with external actors

In Table 4.1, user is involved in all three goals of the pen device. The relevant goals are mapped against relevant interfaces. The term 'mode of deployment' is also introduced in the Table 4.1 which is elaborated in next section.

4.2.2.5 Step 3: Identification of interaction operations in scenarios

In this section following viewpoints, in Figure 4.1a, are discussed and argued:

- Interaction operation;

- Scenario.

There can be many operations in order to achieve a specific goal G1 of a pen device as illustrated in the pen example by Brown & Blessing (2005) in Figure 4.2 and also argued by Vermaas (2009). However, they do not discuss or show associations with relevant interfaces as summarised in Table 4.2.

Table 4.2 Interaction operations description of a pen device

Step 2: Interfaces	**Step 3: Operations**	**Step 1: Goals**
Mode of Deployment M (D,E)	**Interaction operations**	**G1: To provide information**
M (Pen, User)	Operation/Action (O1): Grip the Pen	X
	Operation/Action (O2): Orient the Pen	X
	Operation/Action (O3): Put Pen Tip to Paper	X
	Operation/Action (O4): Apply Pressure	X
	Operation/Action (O5): Move Pen	X
M (Pen, Paper)	Operation/Action (O6): Tip is in contact with paper	X
	Operation/Action (O7): The Tip exerts pressure on the paper	X

Note that all these represent *interaction operations* that exist between pen device and the surrounding or external actors such as user and paper. These operations may occur in a specific mode often referred to as mode of deployment M (D, E) where D stands for device and E for environment (Brown & Blessing, 2005). Brown & Blessing (2005) stated “the mode of deployment is ‘intended to capture the notion of how to use the object so that it produces the intended effect…. It is how you have to hold the device or place it when you need it to function”. Thus it would not be wrong to say that device’s environmental interaction operations exist at interfaces when M (D, E) is established as illustrated in Table 4.2.

Vermaas (2009) call interaction operations as *actions*. Such operational descriptions are also suggested by Daniels & Bahill (2004) and Erisksson et al (2008). A plan consists of such operations. “A **plan** consists of such executable operations or actions which are ordered sequentially or otherwise” (Vermaas, 2009). The executable operations can be thought in sequential or non-sequential order as focus should be on identifying possible interaction operations happening at interfaces. The plan term is equivalent to ***scenario*** in

this thesis, a term adopted from software engineering community (Table 2.5). A scenario can reveal either sequential or non-sequential operations (actions) in an interface to achieve a use case goal as shown below.

Sequential 'main/success scenario' for a pen are illustrated in Table 4.3.

Table 4.3 Sequence based main/success scenario for a pen device

Main/Success Scenario aligned with Plan (Brown & Blessing, 2005) (Figure 4.2) ***Sequential Operations*****.**	
M (Pen, User) between the Pen - User interface.	Grip Pen
	Orient Pen
	Position Pen
	Move Pen
	… etc.

Non-Sequential 'main success scenario' are illustrated in Table 4.4.

Table 4.4 Non-sequence based main/success scenario for a pen device

Non-Sequential Operations.	
M (Pen, User) between the Pen – User interface.	Grip Pen
	Move Pen
	Orient Pen
	Position Pen etc.

There can be both different and common operations across multiple goals e.g. operation 'Move Pen' is common across two goals 'G1: to provide information' and 'G3: to carry through' as illustrated in Table 4.5.

Table 4.5 Interaction operations common across multiple goals

Step 2: Interface	**Step 3: Operations**	**Step 1: Goals**	
Main/Success Scenario: ***Sequential Operations.***		**G1: To provide information**	**G2: To carry through**
M (Pen, User) between the Pen and User	Grip Pen	X	X
	Orient Pen	X	
	Position Pen	X	X
	Move Pen etc.	X	X
M(Pen, Paper) between the Pen and Paper.	Tip is in contact with paper	X	

At this stage, the relevant critical operations belonging to more than one goals of a system can be mapped out and is referred to as first iterative loop in the proposed six-step based interface definition methodology as shown in Figure 4.1b. This fact is not elaborated by Brown & Blessing and by Vermaas either.

There could be alternative or exception e.g. (failure) scenarios (Table 2.5). For the pen example these are illustrated in Table 4.6.

Table 4.6 Alternative or exceptional (failure) scenario for a pen device

Sequential but 'Alternative or Exceptional (Failure) Scenario'	
M (Pen, User) between the Pen and User	Slip Pen,
	…
	Break Pen
	… etc.
M(Pen, Paper) between the Pen and Paper.	Tip creates hole in the paper

From Tables 4.1 to 4.6, it is clear that device's environmental interaction operations (both main success and exceptions related) exist at device ***interfaces***. For example, the interactions presented in Tables 4.2 & 4.5 would exist at user-pen and pen-paper interfaces. This is also aligned with contact and channel model approach (Albers & Zingel, 2011) where Pen being a channelling element and user-pen & pen-paper as two working surface pairs (Figure 2.30).

However, ambiguities can often emerge with some operations descriptions given by Brown & Blessing (2005) and by Vermaas (2009) if interface based thinking is ignored. For example, 'apply pressure' in the pen example in Table 4.2; does this mean 'User effort on Pen' or 'Pen impact on Paper'? Same applies with 'Move Pen'. 'User movies Pen' or 'Pen moves on Paper'? Such descriptions e.g. operation no. 4 & operation no. 5 in Table 4.2 by Brown & Blessing and Vermaas are discussed but not at this level of detail even though an existing device is being analysed. The definition of such operations can be more clear if interface based thinking is adopted, as described in above statements e.g. 'User effort on Pen' etc. Furthermore, 'apply pressure' contains a technical 'noun' description along with 'verb'. At this stage designer is keen to think from subjective domain and not from a technical domain perspective; but if

the designer and stakeholder need overlaps, then such operational statements by a system designer are acceptable. This may lead to a conclusion that verb-noun description of an interaction operation can represent either a subjective or objective statement by the analyst while restricting his/her thinking to subjective domain (stakeholder domain).

Moreover, some operation descriptions by Brown & Blessing and Vermaas are already articulated solutions specific in the mode of deployment M(D, E); e.g. M(Pen, Paper) in Table 4.2 where 'Tip' being in operation descriptions 'Put Pen Tip to Paper' & 'The Tip exerts pressure to paper'. The internal solutions of a device or system should evolve later (at white-box view i.e. one level below) that would deliver solution independent operations (captured at black-box view i.e. one level above) in the systems hierarchy. The 'Tip' is an internal part (or inside actor) of the Pen device (see e.g. nested systems concept by Crilly, 2012). From existing system analysis perspective, these statements by Vermaas and Brown & Blessing are fine but then one should argue that the mode of deployment is solution specific i.e. it would be M (Pen_Tip, Paper) i.e. interface analysis between Pen_Tip (internal actor) & Paper (external actor). As highlighted by Brown & Blessing (2005);

> *"many people have pointed out that in natural language one can describe the function of a device without knowing anything about its structure, or even about exactly what behaviours are at the D to Ei interface".*

Therefore, a methodology is necessary that could aid in making such solution dependent (within the system boundary – white box) and independent (beyond system's boundary – black box) levels' distinction and also could support system-of-systems thinking for both existing as well as innovative systems.

4.2.2.6 Step 4: Identify interaction exchanges

In this section following viewpoints from Figure 4.1a are discussed:

- Interaction operands/exchanges (as behavioural effects)
- Interaction directionality

Note that the interaction operations are often referred to as system behaviour (Section 3.4.2.1.1.1). For example, according to Gedell (2011) "a system's behaviour is the result of system and its interaction with surroundings" (Table

2.9). In this thesis, behaviour view is detailed further when the designer or analyst switches from thinking in stakeholder domain to technical device domain. As presented earlier that the mode of deployment aims to capture the notion of how to use the designed object so that it produces the intended effects to surrounding interacting objects. The intended effects can be referred to responses of a device on its external actors (Figure 2.33 and Table 2.13) or exchange effects (Section 2.4.5.2 and Table 2.8).

Brown & Blessing (2005) stated;

> *"However, to have an "effect" the device needs to behave. When M(D, Ei) is established, some causal interactions between D and Ei are enabled, leading to behaviors. Behaviors can be values of state variables, or relationships between them, either at an instant or over time".*

Vermaas (2010) stated;

> *"Designers and analysts determine which effects behaviour of devices should have in order that the device can play its role in the actions with the device and thus can play its role in realising the goal of the device".*

In Table 4.5, pen behaviour is described from interaction operations perspectives. Following examples in Table 4.7 are given by Brown & Blessing (2005) for behaviours of the pen in its surroundings.

Table 4.7 Behaviours of the pen device

Behaviours: for the Pen Device by Brown & Blessing (2005) (Figure 4.2)	**Author understanding & interpretation: Exchanges as behavioural effects**
Behaviour (B1): ink flows from the tip	Material (M) exchange i.e. ink an ***output*** from the Tip which is an ***internal actor*** of the Pen
Behaviour (B2): ink coats the paper	Material (M) exchange i.e. ink an ***input*** to the Paper which is an ***external actor*** for the Pen
Behaviour (B3): the tip is moving	Tip (i.e. ***internal actor***) changing its behaviour ***over a certain time period***

The behaviour of the pen device on the paper, for instance, may meet conservation laws, but its effect may be described as that pushing a tip leads to punch hole on the paper, where pushing the pen involves a small amount of energy input, and the resulting hole of the paper involves amounts of energy and ink material loss as output (as highlighted by Vermaas, 2009). The relation

between interaction operations and exchanges as behavioural effects are interpreted in Table 4.8.

Table 4.8 Relation between interaction operation and exchanges as effects

	Relation between operations and exchanges as behavioural effects		
Step 2: Interface	**Step 3: Operations**	**Step 4: Exchanges**	**Step 1: Goal**
Mode of Deployment	**Interaction Operations**	**Behaviour (B1, B2,…Bn) as effects**	**G1: To provide information**
M(Pen, Paper) between the Pen and Paper	Pen is in contact with paper	Ink flows from the pen	X
		Ink coats the paper	X
		…	…

In the behavioural examples by Brown & Blessing shown in Tables 4.7 and 4.8, it can be concluded that **effects as behaviour** of device or system can be associated with the (i) ***operands flowing*** in from or out to external/internal actors; as well as (ii) ***changes over a period*** of time for itself or its actors themselves. However, from the examples given by Brown & Blessing it is quite clear that actors and device' behaviours do not change over the time period unless they interact with each other and also that exchanges/operands flow in between them.

For example, a chair on the floor does not change its position on its own over a period of time i.e. *static behaviour*. When user's effort or gust of wind greater than the chair's weight interacts with it, it changes its position over a period of time i.e. a *dynamic behaviour* due to flow of energy, and material operands from user and wind respectively to chair. Thus in dynamic behaviour, exchanges/operands role comes into play.

In static behaviour case, gust of wind less than chair weight does not change its position visibly but may deform its shape or change its structural strength over a period of time e.g. corrosion or wear due to presence of oxygen and moisture contents as exchanges/operands from wind to chair. Thus, it would not be wrong to say that behavioural effects (both: intended and unintended) can be associated and predicted with operands/exchanges based thinking but from the technical domain perspective.

4.2.2.7 Step 5: Assess interaction exchange effect

In this section following viewpoints from Figure 4.1a are discussed;

- System attributes
- Interaction quantification (as criticality scale)

Some interaction exchanges are often considered more important than others for which subjective scales thinking is opted (Figure 2.34), and this design activity is often considered hard to implement in the sense how to identify which interaction is more important than the other. Though this activity is hard and causes hindrance to standard interaction analysis procedure but it helps in prioritising and focusing on the essential requirements than the others. For example, this is quite clear from the arguments and the example given by Helmer (2010);

> *Consider a jet engine's High PressureTurbine (HPT). One stator row is required at the exit of the combustion chamber to divert the fluid flow in circumferential direction. The stator vanes have to be mounted in this position, so that a +2 structural entry is required. There is also a required fluid (i.e. material) flow from the combustion chamber to the stator vanes, which requires adjacency (+2 material). Unfortunately, the vanes are subjected to extremely high heat loads, both due to heat radiation from the combustion itself (consequence:−2 energy mark; the interaction is so detrimental that spatial separation is required, otherwise the vanes would melt) and the hot material flow itself. This extreme example is ideal to show that the +2 marks overrule the −2 mark …, despite the −2 mark, it is obviously possible to solve the problem by applying elaborate and expensive measures (in this case, cooling techniques and choice of material).*

As shown earlier in Table 4.8, the identified exchanges are essential and thus their functional requirements need to be handled carefully and on priority basis. This is often recommended in literature. In this thesis, interactions effect are assessed on engineering attribute types (Figure 2.8) associated with automotive vehicle that involves mechanical, electrical, and software elements as applied to the design and manufacture of motorcycles, automobiles and trucks and their respective engineering subsystems (or modules).

In academic literature, engineering attributes in the context of domain-specific industry (Figure 2.8) is often not discussed explicitly within the existing requirements modelling approaches. It is not discussed how requirements are organised in such multiple attributes. This research does not limit its scope to any domain-specific industry such as automotive industry but the assumption is taken here that any technological system can possess certain high level

attributes. For example, the technological systems such as medical related systems and nuclear power plant system belong to medical industry and nuclear industry and such technological systems can also possess some definitive attributes like vehicle system's engineering attributes (Figure 2.8) that are often considered in automotive companies.

This research expands the requirements modelling research by incorporating this practicality aspect thereby adopting and making analogy with automotive systems attributes. A system engineer or analyst in automotive industry is generally responsible for the delivery of system requirements thereby defining and putting them in relevant attribute types associated with relevant stakeholders: end consumer, governmental regulations, and automobile manufacturer (corporate related) (Ford, 1997). Some examples are listed in Table 4.9.

Table 4.9 Requirements with relevant attributes (adapted from Ford, 1997)

System	Required by Stakeholder (Actor)	Requirement related to Attribute type	Requirement Description
Vehicle	Corporate	A8: Noise, Vibration, Harness (NVH)	***Requirement 1****:* The system shall have a resonant frequency of 40 Hz or greater.
Vehicle	Customer	A3: Accommodation & Usage A1: Safety	***Requirement 2***: The surfaces of controls related knobs shall cause no reflections to driver's or passenger's vision from sunlight and shall meet the safety regulations.

Table 4.9 shows that requirement no. 2 should not affect the customer and it belongs to attributes A1 and A3 of vehicle system. These sort of requirements can be seen as an interface between vehicle system and its stakeholders.

Each attribute can be further broken down and has to be delivered in acceptable limits for which the system engineer often goes through trade-off process. For example, 'A7: performance and fuel economy' is one of the 17-18 attributes (Figure 2.8). End user (one of the stakeholders) may ask for maximum power (i.e. A11: Energy Management) from the engine at the cost of acceptable or standard limit of fuel economy due to regulatory standards. From the company perspective, these are contradictory or opposing requirements of

vehicle system and thus require trade-off decisions. Thus, such opposing requirements can emerge at stakeholders-system interfaces and the impact of interactions happening at interfaces need to be conceptually assessed against system attributes so that relevant and preventive requirements can be derived and grouped upfront in the design process by a system engineer.

Thus, it is assumed in this research that any technological system may possess such engineering attributes that are handled by either different teams or system engineers. Therefore, this thinking is applied to the pen-user interface e.g. as illustrated in Table 4.10.

Table 4.10 Interaction exchange effect assessment

Step 2: Interface	**Step 3: Operations**	**Step 4: Exchanges**			**Step 5: Attributes Impacted**	**Step 1: Goals**
Mode of Deployment M (D,E)	**Interaction Action/ Operation**	**Behaviour as Effects**	**From**	**To**	**Attribute Type & Criticality**	**G1: To provide information**
M(Pen, User) Pen-User Interface	User grips the pen	Human Energy	User	Pen	Accommodation. & Usage (+2)	X
		Warmth Energy	Pen	User	Accommodation & Usage (-1)	X
M(Pen, Paper) Pen-Paper Interface	Pen is in contact with paper	Ink	Pen	Paper	Performance (+2)	X

The interaction exchanges (energy-E, material-M, information-I and spatial/physical-P related) such as human energy (E), human hand (M), and warmth energy (E) may impact the performance of pen device within the interaction operation ‘User grips the Pen’ and thus their impact need to be assessed on pen’s engineering attributes. For example, all such exchanges can influence the ‘accommodation & usage’ attribute of a pen in a desired/undesired way and therefore their criticality can vary and thus five point scale (+2,+1,0,-1,-2) from Pimmler & Eppinger (1994) is adopted (Figure 2.34) for this purpose. For example, as shown in Table 4.10, application of human energy (as effort) in acceptable limit on pen is necessary (i.e. desired input effect), thus has +2 impact on its ‘accommodation & usage’ attribute, whereas warmth from user on pen can be unintended or disturbing effect on device but may not fully

detrimental for achieving a system functionality, thus -1 impact on the attribute and it requires some countermeasure requirement.

4.2.2.8 Step 6: Derive interaction requirements at black-box view

In this section following viewpoints from Figure 4.1a are discussed and argued;

- Functional requirement (as *environment centric function*)
- Non-function performance requirement.

4.2.2.8.1 Step 6a: Specify interaction functional requirement

Based on exchanges as behavioural effects thinking, functional requirements at system interfaces are derived. Therefore, it is envisioned in this thesis that *environment centric functions as* functional requirements (voice of engineer in the technical/objective domain by designer) can be derived from environment centric's interaction operations (voice of customer in the subjective domain by designer) at system interfaces based on operands/exchanges based thinking as a bridge in between them. According to Vermaas (2009), in order to successfully execute the 'use plan' (i.e. specific mode of deployment and operations therein), the analyst (or designer) have to derive the functions that the technical system should deliver. But here arises a question, can we call these *environment centric functional requirements*? The answer to this is first justified with arguments available in Vermaas (2010).

> *Effects of behaviour of a device are now events and processes that are the result of behaviour of the device. These events or processes may consist in states of affairs consisting in the device itself and/or its properties (as in device-centric functions) as introduced by Chandrasekaran and Josephson [10], …**in states of affairs consisting in the environment of the device and/or the environment's properties (as in environment-centric functions), or in states of affairs that are combinations thereof.***
>
> *Effects of behaviour may also be represented by **verb-noun** expression, by evolving state variables or, in the case of processes, by **operations on flows or operands: energy, material, and information. But now the description need not meet physical conservation laws.** The behaviour itself meets those laws, **but the effects may be described in a manner that ignores conservation laws.***

Note that there is a subtle difference between interaction operation (when specified from user/stakeholder perspective) and environment centric functional requirement as intended behaviour as effect (from design perspective) by

designer. For example, in the pen device, an interaction description would be ‘User grips the Pen’. This statement can also be articulated from device perspective but still in stakeholder domain e.g. ‘Pen supports the User Grip’. The former statement expresses external object (i.e. user) does action on a device (i.e. subject) whereas later statement states device (subject) does action on external object (i.e. user). This shows environmental external actor i.e. user (an external object) is involved with the device and the designer can articulate any of such statements by keeping his/her thinking in stakeholder domain.

In response to this action, many behavioural effects can emerge on environmental actors from device or vice versa when designer switches his/her thinking now in technical domain. For example, ‘human energy’ and ‘human hand surface’ as desired inputs as well as ‘human warmth’ as undesired input on pen whereas ‘skin deformation’ would occur on Human as behavioural effects as illustrated previously in Table 4.10. All these effects of behaviour happening between external actor (user) and device (pen) leads to environment centric functional requirements articulated in verb-noun expressions.

Table 4.11 Environment centric functional requirements of a pen

Step 2: Interface	**Step 3: Operations**	**Step 4: Exchanges**			**Step 5: Attributes Impacted**	**Step 6: Environment Centric Requirements**	**Step 1: Goals**
	Interaction Operation	**As Effects & Type**	**From**	**To**	**Attribute Type & Criticality**	**Interaction Functional Requirements**	**G1: To provide information**
Pen-User Interface	User grips the pen	Human Energy (E)	User	Pen	Accommodation & Usage (+2)	Import Human Energy	X
		Warmth Energy (E)	Pen	User	Accommodation & Usage (-1)	Prevent Warmth Energy	X
Pen-Paper Interface	Pen is in contact with paper	Ink (M)	Pen	Paper	Performance (+2)	Export Ink	X

The environment centric functional requirements from device perspective in association with such behavioural effects would be ‘Import Human Energy’, ‘Support Human Hand’, and ‘Sustain Warmth Energy’ etc. that the designer

would like to engineer in the device with respect to its external actor as shown in Table 4.11.

According to Crilly (2012) "…environment-centric functions need only refer to elements external to the device." Thus, in Table 4.11, the derived environment centric functional requirements are associated with external actors i.e. user and paper. Note that one can specify 'Import Warmth Energy' but it may be possible that designer thinks of preventing this warmth as input to pen device and can articulate the preventive function as 'Prevent Warmth Energy' or 'Sustain Warmth/Thermal Energy'. All these statements would be environment centric in the mode of deployment M (Pen, User) derived from interaction operation 'User grips the Pen' taking place between pen-user interface. Similarly, for chair's static behaviour case, thinking of behavioural-effects would result environment centric descriptions 'Prevent Corrosive Material' from air to chair (device).

At this stage M (Pen, User) & M (Pen, Paper) in Table 4.11, pen analysis is purely solution independent whereas external actor's details can be thought or considered as constraints on designed solution. For example, user_hand or user_voice would interact with pen so external actor's details and demands can impose both structural and operational constraints (such as hand contact area, human energy, warmth) on the design choice or in filtering of Pen choices or system designs due to the mode of deployment being external constraint specific i.e. M (Pen, User_Hand) or M (Pen, User_Voice). A designer may like to engineer both modes in a single device, then in that case, deployment model will be M (Pen, User_Hand & Voice) e.g. think of a smart pen device that could operate with hand and voice of user. And if all such constraints with scenarios of an external actor need to be considered with system, then it would be M (Pen, User). Let us compare how environment centric functions (as functional requirements in this thesis) for pen device are given by Brown & Blessing (2005);

Environment-centric function: "to cause a piece of paper to have ink on it" (Figure 4.2)

The above statement holds true as external object (Paper) is affected by device (Pen) having operand (ink as out) and thus environment centric function is between device and external actor. The 'verb' in above environment-centric

function is not clear as such and thus better way of articulating it from pen device perspective would be as ‘Export Ink’ from Device (Pen) to External Object (Paper) as shown earlier in Table 4.11.
The other environment centric functions are also given by Brown & Blessing as follows:

> *Environment-centric function: “to write”;*
> *Environment-centric function: “to communicate information”.*

The above two statements seem not convincing as ‘to write’ indicates what. Where and who does or relate to it e.g. what is the external structure of a device or study context? and which external actors with device achieve this function? Is it between ‘user and device’ or ‘paper and device’ or ‘device and reader’? “Environment-centric functions need only refer to elements external to the device” as discussed by Crilly (2012). Furthermore, the third statement ‘to communicate information to reader’ is the ultimate goal (i.e. to provide information to reader) and not a function.

Note that in this thesis *environment centric functional requirements* are referred to as ***interaction functional requirements*** (see Table 4.11) and are derived from interaction operations based on exchanges/operands as behavioural effects at the interface of *two interacting actors* or between external actor and the system-of-interest. Similarly, interaction functional requirements can also be specified between the interfaces of two *internal* actors of a system.

The interaction functional requirements do not reveal the complete physics of the technical device as evident from Vermaas’ arguments presented earlier as conservation laws are ignored and hence physics is partially covered in them. Thus, it should be noted that in this research environment centric but interaction functional requirement statement is not completely independent of physics.
It is often discussed that a good practice would be if such functions are defined completely independent of physics (Vermaas, 2009), however the examples given by researchers are again not convincing. For example, the lamp’s environment centric function, given by Vermaas (2010), is “to let light fall on objects around the lamp” thereby cloaking the physics, derived from actions such as ‘connect with mains’ and ‘put lamp in the room’. First of all light is not an object rather a result of proton particles travelling in high speed that produce

wavelengths which some are visible to human eyes. Thus such function descriptions then violate basic function definition i.e. ‘verb+object’ at first place. Or alternately and inversely one can argue that function definition based on verb+object (Stone & Wood, 2001) itself is not completely physics independent. However, the focus of this thesis is not on function articulation rather providing a tool that could encapsulate/support any functional description due to different degree of abstraction across system hierarchy as argued by Umeda et al (1996). Secondly, ‘to let light fall on objects’ can be a sub-goal of the lamp for its main goal (i.e. ‘to illuminate the room’) which means that light should fall on the room wall and other things within the room. A main goal can have many sub-goals that can be represented via *include* or *exclude* relationships with external stakeholders (Figure 2.11). On the other hand, it is argued and possible to have physics cloaked by ignoring the exchanges based thinking as reasoned by Vermaas (2010) with above examples (Figure 2.38). Thus, a tool should also support both physics dependent and independent analysis.

From the above function example, given by Vermaas’s (2010), and in existing literature (Stone & Wood, 2001), it becomes apparent that functions are articulated in two ways and can be split into two categories: ‘verb+object’ and ‘verb-non-object’ as shown in Figure 4.5.

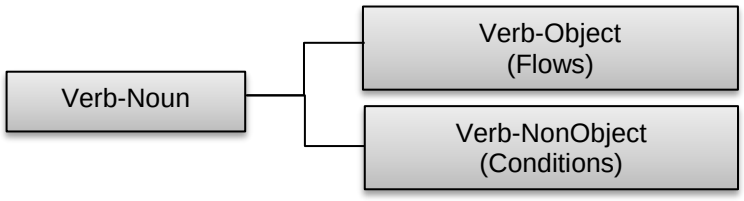


Figure 4.5 Function articulation variants

Functions that follow verb-noun structure but do not contain object (rather condition or result) whereas other contain objects and thus follow noun criteria too. For example, function descriptions in (Figures 2.16 and 2.17), Import Water, Import Solid, Load Bread, Load Dirty_Clothes, Move Pen, Disperse Ink, and Generate Protons; all satisfy both ‘verb-noun’ & ‘verb-object’ structures as descriptions containing flowing objects such as water, solid, bread, dirty_clothes, pen, ink, and protons. On the other hand, function and process descriptions in (Figures 2.14, 2.19, 2.23, and 2.37), Check Out and Display, Maintain Balance, Create Light, Make Sound, Make Collision, Fill Drink in Cup,

and Adjust Brightness etc. only satisfy verb-noun structure as they do not contain flowing objects but rather reflect desired conditions.

Thus, the proposal in this research for function description is to follow **'verb+noun'** structure encapsulating both '**object and non-object**' types as it can be beneficial both from physics cloaking and physics based perspectives as well as consideration of different degrees of abstraction within and across system boundaries as shown in Figure 4.5. Moreover, this will support the fact as not all interaction functional requirements (or functions) can be associated with input-output flowing objects/exchanges only (yet often followed by a group of researchers e.g. Otto & Wood, 2001; Stone & Wood, 2000; Yildirim & Campean, 2014) but also non-input/output related such as surfaces, assembly, standards and conditions related (as approached by other group of researchers e.g. Umeda et al., 1996; Scalice et al., 2008) across other lifecycle phases. For example, in a ball-point pen case, it can be 'Maintain Operational Life' of a pen device.

Note that in this thesis, *interaction functional requirements* result at interfaces due to interaction between two interacting elements (or actors) either inside the system boundary or between the system-of-interest (a conceptualised object) and external interacting actors (objects) beyond its boundary whereas *functions (as solution independent)* reside inside the system boundary dealing with flowing exchanges/flows ***states'*** transitions. Therefore, in this research, following two types of functions are proposed:

- ***Interaction functional requirements*** are derived from interaction exchanges within environment centric operations at system-of-interest's *structural interfaces* (both internally and externally) while;
- ***Transformative functions*** are derived from in/out exchanges' (flows) transitions inside the system's boundary.

In this section, interaction functional requirements have been discussed. The performance specifications also need to be specified to relevant interaction functional requirements which is discussed next.

4.2.2.8.2 Step 6b: Specify non-functional requirement

An interaction functional requirement can be expressed in terms of its measurable (non-functional) performance requirement specifications. A specification should specify the controlled attribute or parameter with constraint

or bounding relations (<, >, = or minimum and maximum), target value, and unit (Figures 2.9 and 2.37c). This defines either the constraint requirement on the functional requirement or on the system. For example, Table 4.12 illustrates the performance specifications of a pen device associated with its relevant functional requirement of 'import human energy' and 'export ink'.

Table 4.12 Performance requirements specification

Step 2: Interface	**Step 3: Operations**	**Step 4: Exchanges**			**Step 5: Attributes Impacted**	**Step 6: Derived Interaction Requirements**		**Step 1: Goals**
	Interaction Operation	**As Effects & Type**	**From**	**To**	**Attribute Type & Criticality**	**Interaction Functional Requirement**	**Interaction Performance Requirement**	**G1: To provide information**
Pen-User Interface	User grips the pen	Human Energy (E)	User	Pen	Accommodation & Usage (+2)	Import Human Energy	User effort (X < KN < Y)	X
		Warmth Energy (E)	Pen	User	Accommodation & Usage (-1)	Prevent Warmth Energy		X
Pen-Paper Interface	Pen is in contact with paper	Ink (M)	Pen	Paper	Performance (+2)	Export Ink	Ink flow rate (X < m^3/sec < Y)	X

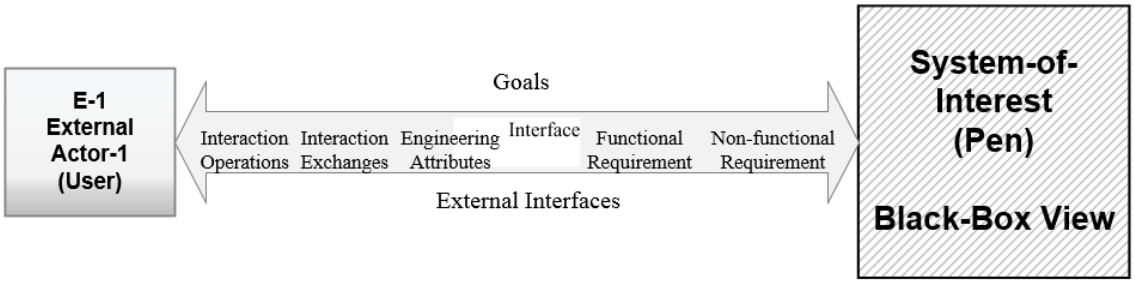


Figure 4.6 Interface definition methodology for external interfaces (black-box)

The discussion on performance aspects is also not highlighted by Brown & Blessing (Figure 4.2) and by Vermaas either. It is also possible that specified requirements may belong to other goals which can be mapped at this stage. For example, 'import human energy' with its performance specification also fits to goal 'to carry through'. This reveals second iterative loop possibility as illustrated earlier in Figure 4.2. Table 4.12 and Figure 4.6 summarises all essential interface definition methodology viewpoints and steps that need to be

considered when defining an interface. Note Table 4.12 in essence represents a structured template for defining and modelling interfaces derived from reasoning with existing available concepts.

4.2.2.8.3 Requirements articulation in traditional style

In order to derive the system requirements in the context of its stakeholders' need and environmental conditions, system requirements are expressed in 'shall' formats (Figure 2.12, and Figure 2.33). The 'shall' format based requirements are often referred to as traditional requirements in literature (Denis & Bahill, 2004). The statement should involve subject and object where subject being the designed system and object being an external interacting object strictly in technical domain so interaction (environment centric) functional requirements can be articulated with traditional formats encapsulating verb-noun format. For example, the interaction functional requirement 'Import Human Energy' from Table 4.12 can be articulated with traditional statement as follows:

> [The pen system shall import human energy] at a [minimum effort of greater than or equivalent to X KN]'.

In similar way, both *interaction functional requirements* and *transformative functions* can be written in traditional 'shall' format. It should also be noted from above statement that measurable constraint and performance conditions are also specified in traditional statements which is often not discussed with the verb+noun format or in descriptive models given by Brown & Blessing and Vermaas. This is one of the reasons that information is often considered lost with verb-noun format as observed by other researchers (Eisenbart, 2014). In the above traditional 'shall' statement's example, the later part reveals the (non-functional) performance aspect (Section 2.4.2.3.2 and Table 2.4) associated with functional requirement. These performance aspects need to be mapped against device's transformative functions (see e.g. Figure 2.19). Therefore, to express the system functionality with its complete information, performance requirement or requirement specification part is also necessary.

4.2.2.8.4 Requirements & exchanges' specifications

It should also be noted that performance specifications of a system associated with functional requirements should not be confused with specifications or

properties of exchanges flowing to/from system. For example, ink has properties like ‘viscosity, density, location, colour’ etc. while interaction functional requirement ‘Export Ink’ for Pen device would have minimum and maximum ‘ink-flow rate’ with or without values as performance specification as illustrated in Table 4.12. Sometimes some specifications may remain common across both exchange and device performance specifications. For example, a cup holder in a car holds a cup and both the cup as an exchange/operand and the cup holder as a device share same geometric specification i.e. diameter. So the functional requirement of a cup holder e.g. ‘Support Cup’ would have performance specification as ‘Min < holding-diameter <Max’ The minimum and maximum values on diameter for cup holder would indicate that it should be capable of holding various types of cup sizes else can support other unintended purposes such as holding bottles, pen, markers etc. Thus, an interface analysis tool should provide a way of distinguishing between exchanges/operands’ properties and system-of-interest’s performance specifications.

So far this section has discussed the steps of proposed interface modelling methodology for capturing system-of-interest’s requirements beyond its boundary (i.e. at black-box). In the next sections, how such methodology should work within system boundary (i.e. at white-box) is also discussed with strong arguments.

4.2.2.9 Derive interaction requirements at white-box view

In this section, following viewpoint is discussed:

- Functional requirement (now as a *device centric function*)

Brown & Blessing (2005) provide following device centric function as an example:

> *Device-centric function: “The function of the pen is to cause ink to flow out of its ink container onto the tip” (Figure 4.2).*

This function description shows the functional relationship between internal components of a pen device. The statement holds true with Crilly’s (2008) work who refers “[…] that device-centric functions refer to the device’s specific structural elements” (Table 2.7).

It is argued here that the given *device centric function description* by Brown & Blessing (2005) and Crilly (2012) is more like ***interaction functional requirement***. As discussed earlier in Section 4.2.2.8.1, the **interaction functional requirement** can be derived based on identified exchanges that exist between two interacting actors. Within the pen device, the interaction operations between the 'Tip' and 'Ink-Container' would be thought first by designer without technical domain thinking and describing them in natural stakeholder language. For example, designers of each actor will think from their own stakeholders' perspective or a system owner from both subsystem designers' perspectives. The ink-container's designer would describe the interaction operation between tip and ink container as 'Ink-Container causes ink to flow onto Tip' whereas ink-container being subject and tip being object. Alternately, the tip's designer may articulate other way round is 'Tip gets the Ink from Container'. The result of such operations during device working operation would be internal behavioural effects on either of two leading 'to cause of ink flow' (a desired effect) or 'ink leakage' (an undesired effect) in between them. This is the stage where designer switches from stakeholder to technical domain thinking and articulate interaction functional requirements in between the two internal actors based on operands/exchanges-based thinking such as material (M) ink related but ***operational*** related i.e. 'Export Ink' from Container and/or 'Import Ink' from Tip perspective as summarised in Table 4.13, and also other interaction functional requirements such as 'Prevent Ink_Leakage' etc.

Table 4.13 Device centric functional requirements of a pen

	Exchange	Device Centric Statements	Internal Actors
Operand Related	*M-Material* *(Ink)*	**OPERATIONAL related** • ***Device Centric - Interaction Functional Requirements: (****Internal solutions based)* *From Container perspective: Export Ink to Tip* *From Tip Perspective: Import Ink from Container*	**Internal Actor 1:** Ink Container - ink flows from. **Internal Actor 2:** Tip – ink flows onto.
		PRODUCTION / INTEGRATION related	
Surfaces Related	P-Physical/ Structural (End Surfaces)	• ***Device Centric - Interaction Functional Requirements: (****Internal solutions based)* *Connect Tip-End to the Ink-Container End* OR *Insert Tip-End to the Ink-Container End*	

It is also mentioned by Brown & Blessing that different relationships can occur during a mode of deployment i.e. ***structural*** related and ***operational*** related (Figure 4.2). So far only the operational related interactions for a pen device have been discussed. The common problem in current approaches is that system operational and structural related functional requirements are often mixed together and kept in same deck. Even if they are put in one deck then interaction classification should be used and is recommended in literature (Figure 2.36 and Figure 2.37d-e) such as E/M/I and P/S in order to differentiate between operational and structural (referred to ***integration/production*** in this thesis) aspects related respectively.

The structural relationships examples given by Brown & Blessing (2005) can be seen in Figure 4.2 and their corresponding device centric functional requirements are articulated in Table 4.13. The structural (P) but ***integration*** and ***production*** related interaction functional requirements can also be specified clearly e.g. 'Connect Tip onto Container', 'Align Tip to Ink_Container' etc. based on interaction operations 'Ink Container holds the Tip' or 'Tip occupies the Ink Container end' respectively.

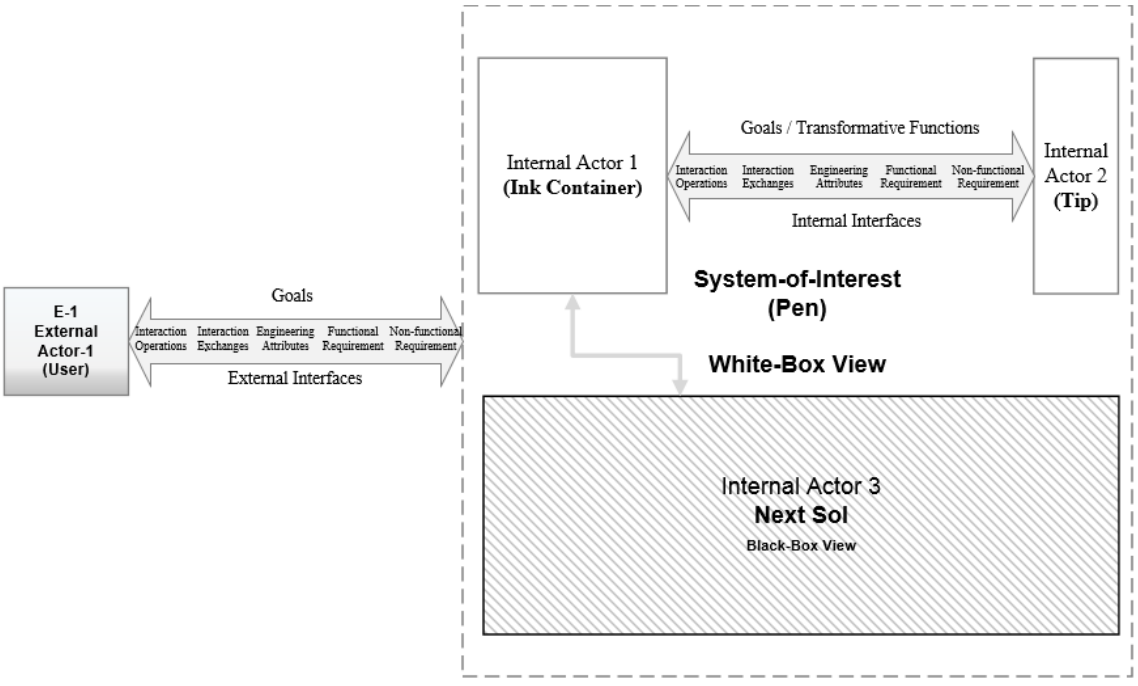


Figure 4.7 Interface definition methodology for internal interfaces (white-box)

Thus, proposed methodology can be applied consistently across black-box and white-box views as illustrated graphically in Figure 4.7. Figure 4.6 shows black-

box view of a pen device with an external interface whilst Figure 4.7 shows its white-box view with internal interfaces and same external interface. It should be noted that only one level of decomposition has occurred in the pen hierarchy e.g. from level 0 to level 1 (Figure 2.7). Each internal actor itself can become a system-of-interest for further analysis e.g. internal actor 3 in Figure 4.7.

Note that unintended effect on external actors may occur due to actual but unintended internal behaviour of a system once its subsystems are synthesised. For example, an unintended effect 'ink leakage' may happen between two internal actors (i.e. ink-container and the tip) within the system boundary (i.e. pen); this would also have consequences as unintended effects across system boundary on its external actors (e.g. user annoyance due to pen causing ink leakage on his hands, or on paper). It is quite possible that such exchanges may not have been thought during design analysis as it can be hard to predict such sort of behaviours within and across system boundary as these often emerge or appear once a system is used or its physical form comes into existence. The other intended behavioural effects such as ink flow if it is achieved without any unintended effects then it is stated that the predicted or expected behaviour of a device (i.e. pen) is achieved.

The device centric function given by Brown & Blessing (2005) in Figure 4.2 is aligned with Chandrasekaran & Josephson (2000) device centric concept as this function involve device's *internal components* thus revealing *internal behaviour* of the device whereas environment centric function reveals the ***device's effects*** on the *environmental (external) actors* (as discussed in earlier sections). Even though, the example given by Brown & Blessing is device centric yet functional statement is *solutions specific* or describe the behaviour within two known internal interacting actors and not ***transformation or solution independent*** functional descriptions that are usually related to flowing operands/exchanges which is recommended by Pahl et al. (2007) and Stone & Wood (2001). These should appear before the internal solutions/actors are searched which provide a functional architecture (i.e. function structure); a fundamental activity in a system architecture analysis (as discussed in Section 2.4.5.2). The internal transformative functions are allocated to internal actors and then subsequently interaction functional requirements are identified as depicted in Figure 4.7. Brown & Blessing (2005) did not discuss this activity as

their focus was on environment and device centric functional descriptions reasoning. Therefore, more detailed discussion on the particular aspect of '(solution independent) transformative function' in this research is presented in Chapter 5.

In the above sections, we have revisited the definition and concepts of various modelling viewpoints that belong to interface definition with the well-known example ball point pen along with other supporting examples.

In the next section, the developed IAT tool is now introduced with its detailed structure based on preliminary derived tabular structure in Table 4.12. Its working is then tested on desktop and industrial case studies.

4.2.3 The interface analysis template (IAT) tool

Based on the reasoning behind the proposed six-step based interface definition methodology (Figure 4.2b), a tabular IAT tool is introduced, as shown in Figure 4.8, to manage the information from those steps of the methodology (in Section 4.2.2). The IAT is more detailed version of Table 4.12 based on thirteen columns C1 to C13.

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Interaction Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Context/Goals
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale / Criticality	Functional Requirement			Nonfunctional Requirement (Performance Related)	Use case desciption
										Verb	Object/Non-Object			
											Input	Output		

Figure 4.8 The Interface Analysis Template (IAT) tool

This research adopted tabular format/template structure for IAT in order to be in line with the current practice as observed from literature (Figure 2.37). The other advantage with IAT's tabular template is that mostly researchers and engineers in academia and industry understand and use tabular templates in contrast to relatively new modelling language such as SysML based tools.

The working steps of IAT tool are similar as discussed in Section 4.2.2. However, the relationships discussed were mainly one-to-one in those viewpoints in previous section with ball-point pen example. On the other hand, it is discussed in Chapter 3 that there can be one to many relationships in between these viewpoints. Therefore, in the next section, a relatively complex electromechanical desktop case study of an electric pencil sharpener is

considered in contrast to pure mechanical based application (ball-point pen) to test the effectiveness of the IAT to manage one-to-many relationships information. The other key aspect to explore is the IAT application across its black-box (Figure 4.6) as well as white-box (Figure 4.7) views at one level of decomposition. Also internal to external interface will be considered and discussed that was not discussed in previous section with pen example.

It is also observed that there are other systems engineering tools that can support IAT. For example, system-of-interest's goals association with relevant external actors can be done via use case diagram (Figure 2.11). Alternately, system context diagram (Figure 2.27) can be used to depict the external structure of a system-of-interest thereby identifying its interfaces with external actors. These two tools were not discussed in previous section in six-step based interface definition methodology. Therefore, these two tools can support IAT and thus are integrated in the interface modelling methodology. In the next section, it is discussed how the IAT in conjunction with systems engineering tools works via an electric pencil sharpener case study.

4.3 IAT Validation: Electric pencil sharpener case study

4.3.1 IAT approach for black-box analysis

To illustrate the deployment of the developed IAT approach, a desktop case study of the design analysis of an electric pencil sharpener is considered at black-box view first.

4.3.1.1 Step 1: Identify goals

A use case diagram is used to show an abstract information by describing the electric sharpener's use cases occurring due to its interactions with external actors such as user, electric energy source, pencil, and environment as shown in Figure 4.9. The box represents the boundary of system-of-interest. In practice, searching actors around a system-of-interest and its goals can lead to difficult discussions but it depends on the project team how and what they define in the problem space.

System engineer responsible for a specific lifecycle phase of a system can develop its use case diagram. For example, Figure 4.9 shows a use case

diagram, for an operational lifecycle phase. A base use case ‘provide sharp pencil’ in turn can have many sub-goals use cases e.g. ‘provide sharp lead pencil’, and ‘provide sharp colour pencil’ that are represented via inclusion and exclusion relationships. Also note that at this stage, engineers can discuss and visualise possibilities of various classes of a single external actor using generalised class usually represented with arrowhead. For example, in Figure 4.9, electric energy source can be DC (direct current) source and AC (alternating current) source.

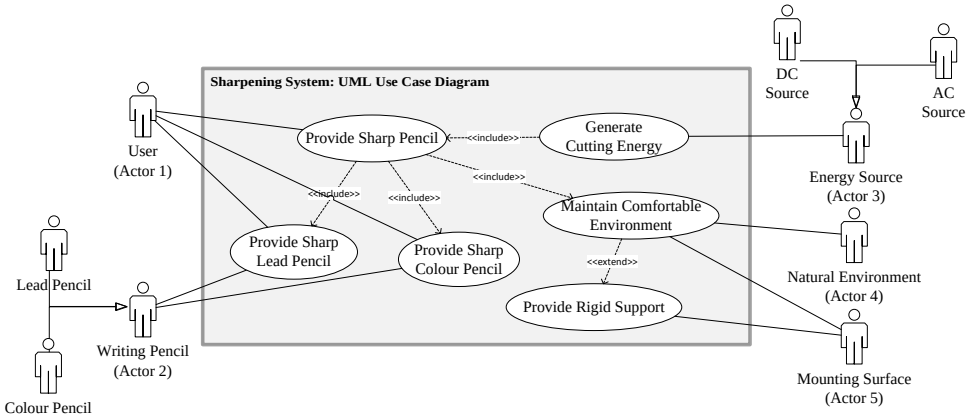


Figure 4.9 Use case diagram for electric pencil sharpener

The identified use cases in use case diagram, either related to the whole system lifecycle or within a specific lifecycle’s phase, can be used as an input in column C13 of IAT, as shown in Figure 4.10.

S1: System Context/Goals
C13
Provide Sharp Lead Pencil

Figure 4.10 Specification of electric pencil sharpener’s goals in IAT

4.3.1.2 Step 2: Identify interfaces

The system context diagram represents the system-of-interest’s external surrounding or environmental boundary. It reflects the overall use cases context analysed via use case diagram (Figure 4.9) and reveals external actors as sources for inputs into the system and destinations of outputs from the system for the achievement of a use case ‘provide sharp pencil’.

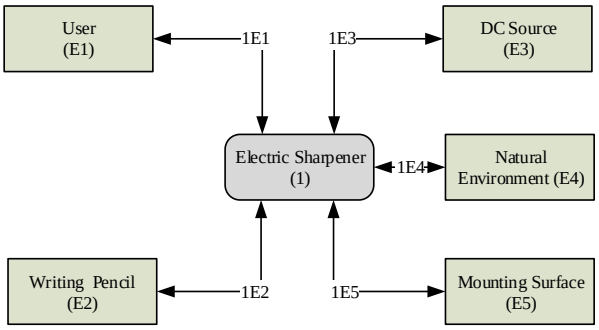


Figure 4.11 System context diagram for electric pencil sharpener

The system context diagram presented in Figure 4.11 describes the black-box functional view of the electric sharpener and contains less information than the 'elaborated context diagram' suggested by Kossiakoff et al. (2011) and Burge (2011) (Figure 2.27a). Labelling each such interaction in the system context diagram between system-actors can lead to unnecessarily complex diagram, and timely process. Therefore, in order to improve the readability of the diagram, it would be better if initially only the distinction between influencing and affecting entities upon each other are made, which is, consistent with the systems theory (Hitchins, 2007). This can be visualised via bidirectional or one directional arrows. For example, in Figure 4.11, an interface '1E1' between an electric sharpener and user represent bidirectional relationship. A single arrowhead would indicate whether an entity affects or affected by the system. The identified interfaces are listed down in column C1 of IAT e.g. for user-sharpener interface, shown in Figure 4.12 along with a relevant use case in C13.

S2: Interface	S1: System Context/Goals
C1	C13
Interface	Provide Sharp Lead Pencil
Electric Sharpener System (1) - User (E1) Interface	X

Figure 4.12 Specification of electric pencil sharpener's interfaces in IAT

4.3.1.3 Step 3: Identify interaction operations in scenarios

Having established the system context and the use cases, the next step is to identify the possible interaction operations in different scenarios between the

external actors and the system. This can be done in following two ways: (i) analyse external actors' one at a time (with or) without sequence of events; or (ii) all actors at simultaneously with sequence of interaction operations in between them. The later strategy is discussed in detail in Section 4.5.

Here, the former strategy is adopted i.e. one actor at a time but without sequences of events as it allows the design team to think both in convergent and divergent manner. Convergent due to the fact that only a single actor's interactions with system is analysed first and divergent due to the fact that all possible interaction operations between that single actor and electric pencil sharpener are analysed. For example, as shown in Figure 4.13, three interactions labelled as rows 1E1-1 to 1E1-3 are identified in column C3 of IAT for 'user-electric sharpener' interface. Similar, procedure can be repeated for other interfaces e.g. electric sharpener and pencil interface.

S2: Interface	S3: Specifications of Interaction Operations		S1: System Goals
C1	C2	C3	C13
Interface	**Interaction Step**	**Operation Description**	**Provide Sharp Lead Pencil**
Electric Sharpener System (1) - User (E1) Interface	1E1-1	User grips the sharpener during sharpening operation	X
	1E1-2	User inserts lead pencil into sharpener	X
	1E1-3	Sharpener alerts the user when sharpening operation is finished	X
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil	X
	...	...	

Figure 4.13 Specification of electric-sharpener's interaction operations in IAT

Iterative Loop 1: If possible, listing interactions in sequential manner would be ideal and this is often recommended by use case modelling for analysing a single use case at a time, whereas IAT also helps to capture and map out multiple interactions that are possible across multiple use cases. For example, having listed down the interaction operations in an interface, two interactions with rows 1E1-1 & 1E1-3 are found to be common across multiple use cases (i.e. for 'provide sharp colour pencil' and 'provide sharp lead pencil') and thus these can even be mapped at this stage, as shown in Figure 4.14. This would highlight which interaction operations are critical and pivotal across system's use cases.

S2: Interface	S3: Specifications of Interaction Operations		S1: System Context/Goals	
C1	C2	C3	C13	
Interface	Interaction Step	Operation Description	Provide Sharp Lead Pencil	Provide Sharp Colour Pencil
Electric Sharpener System (1) - User (E1) Interface	1E1-1	User grips the sharpener during sharpening operation	X	X
	1E1-2	User inserts lead pencil into sharpener	X	
	1E1-3	Sharpener alerts the user when sharpening operation is finished	X	X
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil	X	
	...	...		

Figure 4.14 Mapping of operations against multiple goals of electric sharpener

Alternately, this step of mapping out common interaction operations across multiple use cases can be skipped and a designer can stick to a single use case analysis as illustrated in Figure 4.13.

4.3.1.4 Step 4: Identify interaction exchanges

In this step, each interaction operation identified in step-S3 in column C3 is examined to derive and characterise the flows (and form) related exchange in the next columns. For example, the interaction row 1E1-1 - 'user grips the sharpener during sharpening operation' from Figure 4.13, encompasses two types of interaction exchanges i.e. 'human hand' contact as a physical (P) or material (M) exchange and 'human energy' as energy (E) exchange, as captured in columns C4 and C5 in Figure 4.15.

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Interaction Exchanges					S1: System Context/Goals
C1	C2	C3	C4	C5	C6	C7	C8	C13
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Provide Sharp Lead Pencil
Electric Sharpener System (1) - User (E1) Interface	1E1-1A	User grips the sharpener during sharpening operation	P/M	Human Hand	Hand gripping functional reach (mm)	User	ES	X
	1E1-1B		E	Human Energy	Gripping effort (Pa)	User	ES	
	1E1-2	User inserts lead pencil into sharpener	M	Lead Pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	X
	1E1-3	Sharpener alerts the user when sharpening operation is finished						X
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil						X
	...	...	...	...	...	...	...	

Figure 4.15 Identification of exchanges within electric sharpener's operations

The column C6 in step-S4 of IAT is then used to specify the properties of exchanges. An exchange (or flowing object) can have more than one property as discussed in theory of technical systems given by Hubka & Eder (1988). For example, two key properties of ‘lead pencil’; ‘pencil length and pencil diameter’ are captured in the interaction row 1E1-2 specified in column C6 of IAT tool in Figure 4.15. This column is incorporated in order to differentiate the exchange properties from that of system performance specifications (as discussed in section 4.2.2.8.4). Similarly, all other exchanges’ properties can be specified.
In order to specify the directionality of an identified exchange from/to system-of-interest with its external actors, columns C7-C8 in IAT are used. Also note one can even specify the 'interfaces' in much detail in 'from/to' columns C7-C8 such as 'user_hand' from rows 1E11 to 1E12 and 'user_eye' in row 1E13 instead of only specifying 'user' and similarly for electric sharpener as 'sharpener_surface'.

4.3.1.5 Step 5: Assess interaction exchange effect

The IAT column C9 in step-S5 is used to list down the stakeholders driven system attribute that may be affected by the identified exchange in step-S4 in column C5. For example, in Figure 4.16, in rows 1E1-1A and 1E1-1B, ‘human energy’ and ‘human hand’ exchanges affect ‘accommodation & usage’ design attributes of the electric pencil sharpener.

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Interaction Exchanges					S5: Exchange Effect / Impact		S1: System Context/Goals
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C13
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Provide Sharp Lead Pencil
Electric Sharpener System (1) - User (E1) Interface	1E1-1A	User grips the sharpener during sharpening operation	P/M	Human Hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	X
	1E1-1B		E	Human Energy	Gripping effort (Pa)	User	ES		1	
	1E1-2	User inserts lead pencil into sharpener	M	Lead Pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	A3: Accommodation & Usage	2	X
	1E1-3	Sharpener alerts the user when sharpening operation is finished	I	Sharpening duration	Time (sec)	ES	User	A9: HMI & Audio Visual Performance		X
			I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil	E	Pencil Separation Energy		ES	Pencil	A11: Energy Management	2	X
	...	...	...	...	...	...	...	...	...	

Figure 4.16 Affected attributes identification and assessment against exchanges

Similarly, in row 1E1-3, visual signal affects ‘HMI & Audio Visual Performance’ design attribute. After this, interaction criticality is specified in column C10 with the perspective that how much an exchange (e.g. in row 1E1-A ‘human hand’ exchange in column C5) is critical in relation to a design attribute (e.g. ‘Accommodation & Usage’). The five point scale from +2, +1, 0,-1,-2 is used in to scale each exchange effect on its corresponding attribute. The +1 with A3: Accommodation & usage in row 1E1-A shows that interaction exchange is beneficial but not essential for the achievement of a goal ‘provide sharp lead pencil’. The same scale is applied for other exchanges and attributes.

4.3.1.6 Step 6: Derive system requirements

4.3.1.6.1 Step 6a: Specify interaction functional requirements

There may be one or many interaction functional requirements associated with each interaction exchange and thus these are derived based on exchanges descriptions as shown in Figure 4.17. For example, in user-electric sharpener interface, row ‘1E1-1’ has got two interaction functional requirements in column C11 of IAT i.e. 'support human hand' and 'import human energy' which are derived based on exchange descriptions in column C5 of IAT. It is also possible with an exchange to specify multiple functional requirements e.g. in row 1E1-2 in ‘electric sharpener-user’ interface, an exchange ‘lead pencil’ in turn requires two functional requirements to be specified i.e. ‘import lead pencil’ and ‘secure lead pencil’ as captured in IAT in Figure 4.17.

Note that the interaction functional requirements need to be written from sharpener’s perspective (which is a system-of-interest). The requirements in column C11 of IAT can even be articulated in traditional ‘shall’ statements e.g. 'The electric sharpener system shall support user/human hand' in C11 of IAT for row 1E1-1A. It can also be seen in Figure 4.17 that system requirements may fall into both input-output (such as 1E1-B and 1E1-2) and non-input/output categories (such as rows 1E1-1A and 1E13).

S2: Interface	S3: Specifications of Interaction Operations			S4:Specification of Interaction Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Context/Goals	
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13		
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Functional Requirement Verb	Object/Non-Object Input	Output	Nonfunctional Requirement (Performance Related)	Provide Sharp Lead Pencil	Provide Sharp Colour Pencil
Electric Sharpener System (1) - User (E1) Interface	1E1-1A	User grips the sharpener during sharpening operation	P/M	Human Hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand		Contact surface area (X mm ^ 2 < A < Y mm ^ 2)		
													Contact surface static friction (X< u <Y)	X	X
	1E1-1B		E	Human Energy	Gripping effort (Pa)	User	ES		1	Import	Human Energy		Human gripping effort (XN < F < YN)		
	1E1-2	User inserts lead pencil into sharpener	M	Lead Pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	A3: Accommodation & Usage	2	Import	Lead Pencil		Intake diameter for pencil (X mm < D < Y mm)		
													Ideal point intake depth (X mm < d < Ymm)	X	
										Secure	Lead Pencil		Intake pencil positioning (XYZ)		
	1E1-3	Sharpener alerts the user when sharpening operation is finished	I	Sharpening duration	Time (sec)	ES	User	A9: HMI & Audio Visual Performance		Complete	Sharpening operation		Sharpening completion time (X sec < t)		
			I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	Diplay		Visual signal	Luminous flux (X lumens ≤ Lumens)	X	X
	...	...	...	...	...	...	...	...	...	...	...	...	...		
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil	E	Pencil Separation Energy		ES	Pencil	A11: Energy Management	2	Transfer		Separation Energy	Cutting energy (X ± y W-s/mm ^ 2)	X	
	...	...	...	...	...	...	...	...	...	...	...	...	...		

Figure 4.17 The IAT tool for electric sharpener: derived requirements through interface analysis

4.3.1.6.2 Step 6b: Specify non-functional requirements

Using the interaction operations and their exchanges as the basis for identifying the interaction functional requirements; IAT then helps in identifying the key performance characteristics associated with each functional requirement for an

interface. There can be many (non-functional) performance requirements associated with the single interaction functional requirement. For example, in Figure 4.17, in row 1E1-2, for an interaction functional requirement ‘import lead pencil’ in column C11; two key performance specifications are specified with it for the electric pencil sharpener: (i) the intake diameter for pencil (X mm < d < Y mm) and (ii) the ideal point in-depth (mm) for pencil’s ideal point creation. These performance specifications associated with functional requirements of a system are operational aspect related.

There can be structural/production related performance requirements as discussed in Section 4.2.2.9 and those can also be captured in IAT. Note that the non-functional requirements on the whole system can be written in the language or format of functional requirements. For example, a structural related but performance requirement coming on electric sharpener through its interface with ‘packaging compartment’ actor would be as follows;

> ‘The electric sharpener system shall occupy space into packaging compartment or box’;

which in essence is non-functional requirement but written in verb-noun language/format i.e. ‘support/occupy space’ with its performance specification of attribute and measurable unit as follows;

> ‘with enclosing volume or space (mm x mm x mm or mm^3)’.

The ‘enclosing space’ can be described as a physical exchange (P) that shows a physical constraint on the system coming from its external actor.

Loop 2: Once all requirements are derived by analysing a single use case as shown in Figure 4.17 then these need to be checked across multiple use cases that will help the designer to map out common or prioritise shared requirements as can be observed in Figure 4.18. Thus, mapping of common operations (loop 1) and common requirements (loop 2) across multiple use cases are possible at two stages of the IAT tool which gives it advantage other use case driven or interaction driven approaches. In this way, the IAT can be applied for defining the system-of-interest requirements with its all environmental interfaces.

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Interaction Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Context/Goals	
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13	
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Functional Requirement Verb	Object/Non-Object Input	Output	Nonfunctional Requirement (Performance Related)	Provide Sharp Lead Pencil	Provide Sharp Colour Pencil
Electric Sharpener System (1) - User (E1) Interface	1E1-1A	User grips the sharpener during sharpening operation	P/M	Human Hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand		Contact surface area (X mm ^ 2 < A < Y mm ^ 2)	X	X
													Contact surface static friction (X< u <Y)		
	1E1-1B		E	Human Energy	Gripping effort (Pa)	User	ES		1	Import	Human Energy		Human gripping effort (XN < F < YN)		
	1E1-2	User inserts lead pencil into sharpener	M	Lead Pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	A3: Accommodation & Usage	2	Import	Lead Pencil		Intake diameter for pencil (X mm < D < Y mm)	X	
													Ideal point intake depth (X mm < d < Ymm)		
										Secure	Lead Pencil		Intake pencil positioning (XYZ)		
	1E1-3	Sharpener alerts the user when sharpening operation is finished	I	Sharpening duration	Time (sec)	ES	User	A9: HMI & Audio Visual Performance		Complete	Sharpening operation		Sharpening completion time (X sec < t)	X	X
			I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	Diplay		Visual signal	Luminous flux (X lumens ≤ Lumens)		
Electric Sharpener System (1) - Pencil (E2) Interface	1E2-1	Sharpening system sharpens the pencil	E	Pencil Separation Energy		ES	Pencil	A11: Energy Management	2	Transfer		Separation Energy	Cutting energy (X ± y W-s/mm ^ 2)	X	
	...	...	...	...	...	...	...	...	...	...	...	...	...		

Figure 4.18 The IAT for electric sharpener: derived requirements mapping against multiple use cases

In this section, it is seen that IAT is applied on system's black-box view (i.e. beyond system boundary). In the next section, it would be seen how IAT captures the white-box view information of the same system where the interfaces need to be defined between subsystems and thus derived requirements should be articulated from both subsystems' perspectives.

4.3.2 IAT approach for white-box analysis

IAT tool has been discussed so far with its working on external interfaces at system's black-box view (i.e. beyond system boundary). It is also envisioned that IAT can also be applied for defining and analysing internal interfaces between subsystems at system's white-box (i.e. within system boundary). An electric pencil sharpener is composed of its internal actors (i.e. subsystems/modules) such as pencil fixture, DC motor, and gear train etc. Two interfaces i.e. internal to internal actors interface (DC motor – pencil fixture) and internal actor to external actor (pencil fixture – lead pencil) interface are considered to test the IAT tool with same structure and interface modelling viewpoints.

Once a physical architecture for an electric sharpener is chosen out of many architectures or concepts based on trade-off criteria, then interfaces within selected concepts or subsystems are defined, and detailed. Figure 4.19 shows two internal actors (subsystems) of an electric sharpener with an external actor – pencil. This can be described as an electric sharpener's system boundary diagram.

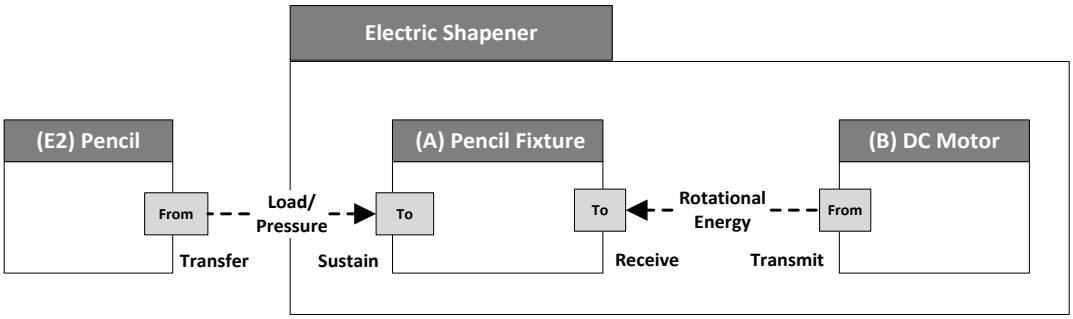


Figure 4.19 Electric sharpener boundary diagram with two internal & one external actors

Initially, at black-box (Section 4.4.1), all requirements were written from only system's perspective but at its white-box, the design team has to switch off one sided view thinking in C11 of IAT due to subsystems all being system-of-interest on their own. This means at white-box, focus is not on one particular subsystem but identifying requirements at the interfaces of both interacting subsystems, and thus both sided perspectives are equally important to capture. Interactions between two internal actors/subsystems are analysed by following directionality rule, i.e. 'from-to' thinking i.e. what goes 'from' where 'to' where which is important at this level of system decomposition (at white-box).

For example, Figure 4.19 shows that an exchange of 'rotational energy' occurs *from '*DC-motor' *to* 'pencil fixture'. From DC motor's perspective (DC system's sided view), interaction functional requirement associated with this exchange would be 'transmit rotational energy' whereas from pencil fixture's perspective (pencil fixture's sided view), the requirement would be 'accept/receive rotational energy'. Now how these both perspectives are captured in IAT tool? This is shown in Figure 4.20. Once the operation (columns C2 to C3) and exchange descriptions (C4 to C8) are specified then the pair-wise requirements from both subsystems perspective are written in column C11 as can be seen in row AB-1. Note that the directionality 'from (DC motor) – to (pencil-fixture)' remains same from both subsystems perspective with respective to an exchange as evident in columns C7-8 in Figure 4.20. Similarly, requirements between an external actor to internal actor (subsystem) can also be listed as illustrated in Figure 4.20 in row E2A-1.

S2: Interface	S3: Specifications of Interaction Operations			S4:Specification of Exchanges					S5: Exchange Effect / Impact			S6: Interface Requirements between two interacting actors				S1: System Context
C1	C2	C3		C4	C5	C6	C7	C8	C9	C10		C11			C12	C13
Interface	Interaction Step	Operations	Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale		Functional Requirement Verb	Object/Non-Object Input	Object/Non-Object Output	Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
Pencil Fixture (PF) (A) - DC Motor (DCM) (B) Interface	AB-1 DCM sided view	Mechanical energy need to be exchanged between Pencil fixture and DC motor		E	Rotational energy	Angular velocity (rad/sec) Torque (Nm)	DCM	PF	A11: Energy Management	2		Transmit		Rotational energy	Angular velocity (X<rad/s<Y) Torque (X<Nm<Y)	X
	AB-1 PF sided View											Accept	Roational Energy		Angular velocity (X<rad/s<Y) Torque (X<Nm<Y)	X
Lead Pencil (LP) (E2) - Pencil Fixture (PF) (A) Interface	E2A-1 LP sided view	Lead Pencil is inserted inside Pencil Fixture		E	Pencil loading impact	Loading impact (Pa)	LP	PF	A1: Safety	1		Transfer		Loading impact	Loading impact (X Pa)	X
	E2A-1 PF sided view	Pencil Fuxture intakes lead pencil										Sustain	Loading impact		Loading impact (X Pa)	X
	E2A-3	Pencil fixture rotates around lead pencil		E	Rotation energy	Angular velocity (rad/sec)	PF	LP		0		Transfer		Rotational energy	Angular velocity (rad/sec)	X

Figure 4.20 IAT for electric sharpener analysis at white-box

Following two key points of IAT tool should also be noted:

- The ‘interaction operation’ description (voice of customer) in column C3 may remain same from both internal actors/subsystems perspectives (see in Figure 4.20 e.g. rows AB-1 & column C3). Column C3 description can also differ for both subsystems (see e.g. row E2A-1 & column C3).

The interaction operation description e.g. ‘mechanical energy need to be exchanged’ in column C3 may remain same but interaction functional requirements description (such as transmit & accept) would differ for both subsystems as illustrated in column C11 of row AB-1 in Figure 4.20.

- The other key point to note is that if interaction operation description in C3 overlaps with exchange description in C5 (see e.g. row AB-1 in Figure 4.20) then it means ‘operation description’ is an ‘exchanged based’ and thus both can be used interchangeably and hence any one column (either C3 or C5) can be left empty.

The clarity, relationship, and articulation of descriptions among the *interaction operation, exchange*, and *functional requirement* viewpoints is often not seen in existing ICDs (tabular) or other interface modelling templates (graphical such as sequence diagram).

Thus IAT, can be applied in similar way in systems hierarchy. As an example, think of applying the IAT tool by zooming-in into the DC motor system’s subsystems (such as armature and shaft components) in Figure 4.19. Therefore, this proves the fact that without changing the order of structure of modelling viewpoints of IAT tool; it can be applied uniformly and robustly from black-box to white-box views of a system.

4.4 Discussion for IAT use in different design scenarios

In this section, the guidelines for the IAT tool usage are discussed in the context of innovative design and lessons learnt activities with same electric pencil sharpener case study. Along with this, it is also presented that IAT is flexible enough to accommodate various sorts of diverse working styles.

In engineering design, for the development of a system with new features, design engineers mostly consider desired interactions and not often detrimental interactions. The reason being that detrimental interactions are discovered late once the decisions on design solutions are made or its physical form comes into existence. They basically appear (detrimental behavioural interactions) when design solutions are synthesized and tested. Therefore, keeping the information of those interactions via IAT tool can be a good practice for future systems design i.e. it will support reverse and redesign activities. In reverse engineering,

and redesign scenarios, an existing physical system is analysed to see how it works, what went good and wrong, and how it could be improved further.

4.4.1 Use of IAT tool for innovative design

In innovative design, the main emphasis is on identifying the system's desired interactions (than undesired or detrimental), which can be worked in following two ways; (i) sequence of interactions or (ii) without sequence of interactions. Let us see how IAT can be worked out in innovative design using the aforementioned two ways.

4.4.1.1 Sequence based interactions in main success scenario

It is possible that engineers often think of interaction operations as actors' actions on the system and then responses of a system in reaction to those actors' actions (Eriksson et al, 2008). The physics behind this is more like Newton's 3rd law that every action (e.g. desired or undesired actions from actors) has equal but opposite reaction (e.g. desired or undesired responses from system). Once a system's use case is listed along with relevant actors then each system-actor interface can be worked out by thinking of sequence of interactions.

S2: Interfac	S3: Specifications of Interaction Operations		S4:Specification of Interaction Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Operation Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Functional Requirement			Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
										Verb	Object/Non-Object Input	Object/Non-Object Output		
	Main Success Scenario Or Desired Interactions													
ES system (1) - User (E1) Interface	1E1-2	User inserts pencil into sharpener during operation	M	Lead pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	A3: Accommodation & Usage	2	Import	Lead Pencil		Intake diameter for pencil (X mm < D < Y mm)	x
													Ideal point intake depth (X mm < d < Ymm)	x
	1E1-5	Sharpener alerts the user when sharpening operation is finished	I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	Diplay		Visual signal	Luminous flux (X lumens ≤ Lumens)	x
													Visible spectrum (X mm < wavelength < Ymm)	x
	...	...	...	...	...	...	...	...	...	...	...	...	...	
ES system (1) - Lead Pencil (LP)(E2) Interface	1E2-3	Sharpening system detects that lead pencil entered	M	Pencil quality type	Pencil surface quality 1	ES	LP	A2: Security	2	Detect	Pencil type		Pencil material detection	x
													Pencil type recognition duration (sec)	x
	1E2-4	Sharpening system starts sharpening the pencil	E	Pencil Separation Energy		ES	LP	A11: Energy Management	2	Transfer		Separation Energy	Cutting energy (X ± y W-s/mm ^ 2)	x
	...	...	...	...	...	...	...	...	...	...	...	...	...	

Figure 4.21 Sequential operations based IAT with main success scenario

For example, in Figure 4.21, for electric pencil sharpener's use case 'provide sharp pencil', design analysis begins with interaction operation 1E1-2 description in column C3, e.g. 'user inserts pencil'. The system in response to this action, does response with interactions1E2-3 and 1E2-4 occurring simultaneously with lead pencil, and then gives final response back to user as an interaction 1E1-5 (column C3). These all are desired interactions (actions &

responses) in top-down sequential path. Once interaction operations are specified in sequential order in column C3, then rest of the columns can be detailed.

It should be noted no undesired interactions (actions or responses) are identified here as it can be hard at this stage of innovation design to identify what can go wrong. It is also mentioned on one occasion in literature that once a design or conceptual design is developed then investigation of undesirable possible actions or responses via only affordance-based theory is achievable (Brown & Blessing, 2005). As discussed in Chapter 2 (Section 2.4.11.1.2), affordances are “the set of interactions between artefact and user in which properties of the artefact are or may be perceived by the user as potential uses” (Maier & Fadel, 2003). Brown & Blessing (2005) also presented that affordance based reasoning is more like a given a device predict possible user actions. This leads to the fact that how one can identify undesired interactions before designing it.

4.4.1.2 Non-sequence based interaction operations

It is quite possible that engineers may opt to think of all possible interaction operations of external actors with system first in non-sequential order. It means there is no need to think of sequential operations occurring between a single and multiple interfaces. For example, in Figure 4.22, in a sharpener-user’s interface, after specifying 1E1-1 interaction in C3 in step 3, engineers can move across other viewpoints to think of exchanges in it (in C5) and thereafter until the interaction functional requirement and non-functional requirement in C11 and C12 and then mapping out across multiple use cases.

S1: Interface	S2: Specifications of Interaction Operations		S3:Specification of Exchanges					S4: Exchange Effect / Impact		S5: System of Interest Requirements				S6: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Operations Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Criticality	Functional Requirement			Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
										Verb	Object			
											Input	Output		
ES system (1) - User (E1) Interface	1E1-1	User grips the sharpener during sharpening operation	P/M	Human hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand		Contact surface area (X mm ^ 2 < A < Y mm ^ 2)	X
													Contact surface static friction (X< u <Y)	X
			E	Human energy	Gripping effort (Pa)	User	ES	A9: HMI & Audio Visual Performance	1	Import	Human Energy		Human gripping force (XN < F < YN)	X

Figure 4.22 Non-sequential operations based IAT with desired interactions

However, in this case sequential thinking (i.e. top-down thinking) is lost within interaction operation viewpoint but here abstract to detail information is worked out first (i.e. from left to right C3 to C13). And then same procedure is repeated

for the next row with definition of interaction operation as illustrated in Figure 4.23.

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Operations Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Criticality	Functional Requirement			Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
										Verb	Object/Non-Object Input	Object/Non-Object Output		
ES system (1) - User (E1) Interface	1E1-1	User grips the sharpener during sharpening operation	P/M	Human hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand		Contact surface area (X mm ^ 2 < A < Y mm ^ 2)	X
													Contact surface static friction (X< u <Y)	X
			E	Human energy	Gripping effort (Pa)	User	ES	A9: HMI & Audio Visual Performance	1	Import	Human Energy		Human gripping force (XN < F < YN)	X
	1E1-2	User inserts pencil into sharpener during operation												

Figure 4.23 Abstract to detail information generation row-by-row basis

4.4.2 Use of IAT tool for iterative design or lessons learnt

In engineering design, the emphasis is often on improving the existing design thereby utilising the existing knowledge/data regarding what went good (i.e. looking at achieved desired interactions) and also by documenting regarding what went wrong (i.e. detrimental interactions that external actors suffer from) at interfaces that were not captured before. Again this can be worked in following two ways; (i) interactions with sequences or (ii) without sequence of interactions. Let us see how IAT can be worked out in the case of improving an existing design.

4.4.2.1 Sequence based interactions

Let us assume that design team developed electric pencil sharpener based on captured information in IAT in Figure 4.21. User buys and uses the electric pencil sharpener and finds problems with it. For example, designer is informed that user used the electric sharpener and gave negative feedback on it with statement ‘sharpener releases wastes or debris during sharpening and dirty my hands’. The designers can utilise this feedback and analyse it by articulating the interaction operation as ‘User finds wastes during sharpening operation while gripping it’ in IAT in column C3 labelled as 1E1-1a and puts it in ‘alternate flow (or failure) scenario’ as illustrated in Figure 4.24. The designer relates this to relevant sequence based interaction operation specified previously during conceptual design phase. For example 1E1-1 can be associated with this undiscovered behavioural operation, thus labelled with alphabet ‘a’ revealing the fact that this undesired behaviour may have occurred during the execution of relevant operation 1E1-1 of the system.

S1: Interface	S2: Specifications of Interaction Operations		S3:Specification of Exchanges					S4: Exchange Effect / Impact		S5: System of Interest Requirements			S6: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11		C12	C13
Interface	Interaction Step	Operations Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Critical	Functional Requirement Verb	Object Input / Output	Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
ES system (1) - User (E1) Interface	**Main Success Scenario Or Desired Interactions**												
	IE1-1	User grips the sharpener during sharpening operation	P/M	Human hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand	Contact surface area (X mm ^2 < A < Y mm ^2)	X
			E	Human energy	Gripping effort (Pa)	User	ES	A9: HMI & Audio Visual Performance	1	Import	Human Energy	Contact surface static friction (X< u <Y)	X
	IE1-2	User inserts pencil into sharpener during operation	M	Lead pencil	Pencil diameter 1 (mm) Pencil ideal length 1 (mm)	User	ES	A3: Accommodation & Usage	2	Import	Lead Pencil	Human gripping force (XN < F < YN)	X
												Intake diameter for pencil (X mm < D < Y mm)	X
												Ideal point intake depth (X mm < d < Ymm)	X
	IE1-5	Sharpener virtually alerts the user when sharpening operation is finished	I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	Display	Visual signal	Luminous flux (X lumens ≤ Lumens)	X
												Visible spectrum (X mm < wavelength < Ymm)	X
	Exceptional Scenario or Undesired Interactions												
	IE1-1a	User grips the sharpener but finds waste material	M	Graphite Debris	Debris waste (mg)	ES	User	A3: Accommodation & Usage	-1	Seal	Graphite Debris	Debris sealing ip	X
	IE1-1b	Sharpener exerts mechanical shock on user's grip during operation	E	Mechanical Vibration	Vibrational energy (J)	ES	User	A8: Noise, Vibration & Heat A3: Accommodation & Usage	-1	Reduce	Vibrational Motion	Vibrational energy (Vibrations <YJ)	X
	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 4.24 Sequential operations based IAT with main success & exceptional scenarios

Note the difference between Figures 4.21 and 4.24, putting the interactions in different scenario's classes (such as desired/main success and undesired or exceptional) can also help the engineers in the sense that in which sequence of event or stage of interaction, undesired interactions emerge or appear that lead to unintended behaviour of a system. Based on this, the relevant exchanges and interaction functional and performance requirements are specified in order

to make design more robust, safer and easy to use. For example, in the above case of what went wrong, a countermeasure interaction functional requirement is specified i.e., ‘Seal debris/graphite’ in row 1E1-1a in IAT column C11 in Figure 4.24. Hence, IAT supports the innovative and re-design engineering activities.

4.4.2.2 Interaction with non-sequence of events

Using Figure 4.23 with non-sequential interactions, a system can be improved based on gained feedback from customers. In this case, there is no need to put and document the detrimental interaction in the separate scenario class. For example, in Figure 4.25, interaction 1E1-1a is specified right underneath the affected interaction 1E1-1. Once, it is documented, then necessary preventive interface requirements can be identified and mapped across multiple use cases.

S2: Interface	S3: Specifications of Interaction Scenarios		S4:Specification of Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Interaction Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Critical	Functional Requirement			Nonfunctional Requirement (Performance Related)	Provide Sharp Pencil
										Verb	Object/Non-Object Input	Output		
ES system (1) - User (E1) Interface	1E1-1	User grips the sharpener during sharpening operation	P/M	Human hand	Hand gripping functional reach (mm)	User	ES	A3: Accommodation & Usage	1	Support	Human Hand		Contact surface area (X mm ^ 2 < A < Y mm ^ 2)	x
													Contact surface static friction (X< u <Y)	x
			E	Human energy	Gripping effort (Pa)	User	ES	A9: HMI & Audio Visual Performance	1	Import	Human Energy		Human gripping force (XN < F < YN)	x
	1E1-1a	User grips the sharpener but finds waste material	M	Graphite Debris	Debris waste (mg)	ES	User	A3: Accommodation & Usage	-1	Seal		Graphite Debris	Debris sealing Ip	x
	1E1-2	User inserts pencil into sharpener during operation												
	1E1-5	Sharpener virtually alerts the user when sharpening operation is finished	I	Visual signal	Luminous flux (lumens)	ES	User	A9: HMI & Audio Visual Performance	2	Diplay		Visual signal	Luminous flux (X lumens ≤ Lumens)	x
													Visible spectrum (X mm < wavelength < Ymm)	x
	1E1-2a	Sharpener exerts mechanical shock on user’s grip during operation	E	Mechanical Vibration	Vibrational energy (J)	ES	User	A8: Noise, Vibration & Heat A3: Accommodation & Usage	-1	Reduce		Vibrational Motion	Vibrational energy (Vibrations <YJ)	x
	...	...	...	...	...	...	...	...	...	...	...	...	...	

Figure 4.25 Non-sequential operations based IAT with undesired interactions

4.4.3 Key points

Following four key points become clear from the previous section:

- The IAT, in Figure 4.8, can be approached in two ways from documentation perspective: either horizontally (left-right) or vertically (top-down) from column C3 to C12.
  For example, on one hand in horizontal direction (Figure 4.23), having identified an interaction in step-S3 e.g. in row '1E1-1' in column C3, then at the same time all other columns across steps-S4, S5 and S6 can be filled step by step (from left-to-right) for a same single row, before listing down all the interaction operations in step-S3 in column C3. This means

increase in incremental knowledge from abstract to detail on row-by-row basis.

On the other hand, on the vertical deployment (Figures 4.13 and 4.15), all interactions in step 3 in column C3 are listed first (in top-down manner) and then other columns from step 4 till 6 are filled one after another. This means increase in incremental knowledge on column-by-column basis.

- The IAT can also be approached in two ways from design analysis perspective: sequential and non-sequential based thinking. When IAT is used with sequence based operational thinking; the row-by-row or column-by-column styles can be adopted therein. Similarly, when IAT is used with non-sequential based operation thinking, any one of the two documented styles can be adopted.
- It is also shown that IAT tool is consistently implementable both across system's black-box and white-box views for deriving requirements.
- The IAT structure remains same for defining and analysing both external (Figure 4.17) and internal interfaces as well as internal to external interfaces (Figure 4.20).
- IAT provides more structured and complete detailed information due to robust integration of diverse but essential interaction modelling viewpoints.
- It is also possible that two viewpoints can overlap (e.g. operation description i.e. Column C3 as voice of customer and exchange description i.e. Column C5 as voice of engineer) and in that case one viewpoint can be left empty.

In the next section, relationships between interface modelling viewpoints for a system-of-interest are summarised and represented via UML class diagram.

4.5 Formalised model of IAT

This section provides a formalised model of the methodology of IAT via Unified Modelling Language (UML). The UML is a visual, object-oriented, and multi-purpose modelling language (Engels et al. 2005). It is a modelling language that has been used for specifying the qualitative knowledge to represent the systems design (Bartolomei, 2007; Devanathan & Ramani, 2011). It provides a

formalised way to write a system model, covering conceptual ideas (Breu et al. 1998). It can be used to account for relationships between heterogeneous concepts or views of a process or methodology. Here, UML diagram is used to formalise an information model of interface definition methodology and IAT which summarises the integration of various viewpoints along with clear relations between them.

The ball-point pen and electric pencil sharpener desktop case studies have given the enough evidence for summarising the one-to-one and one-to-many relationships among the various modelling viewpoints of a system's interface which are now addressed and represented via a formal modelling language UML class diagram as shown in Figure 4.26.

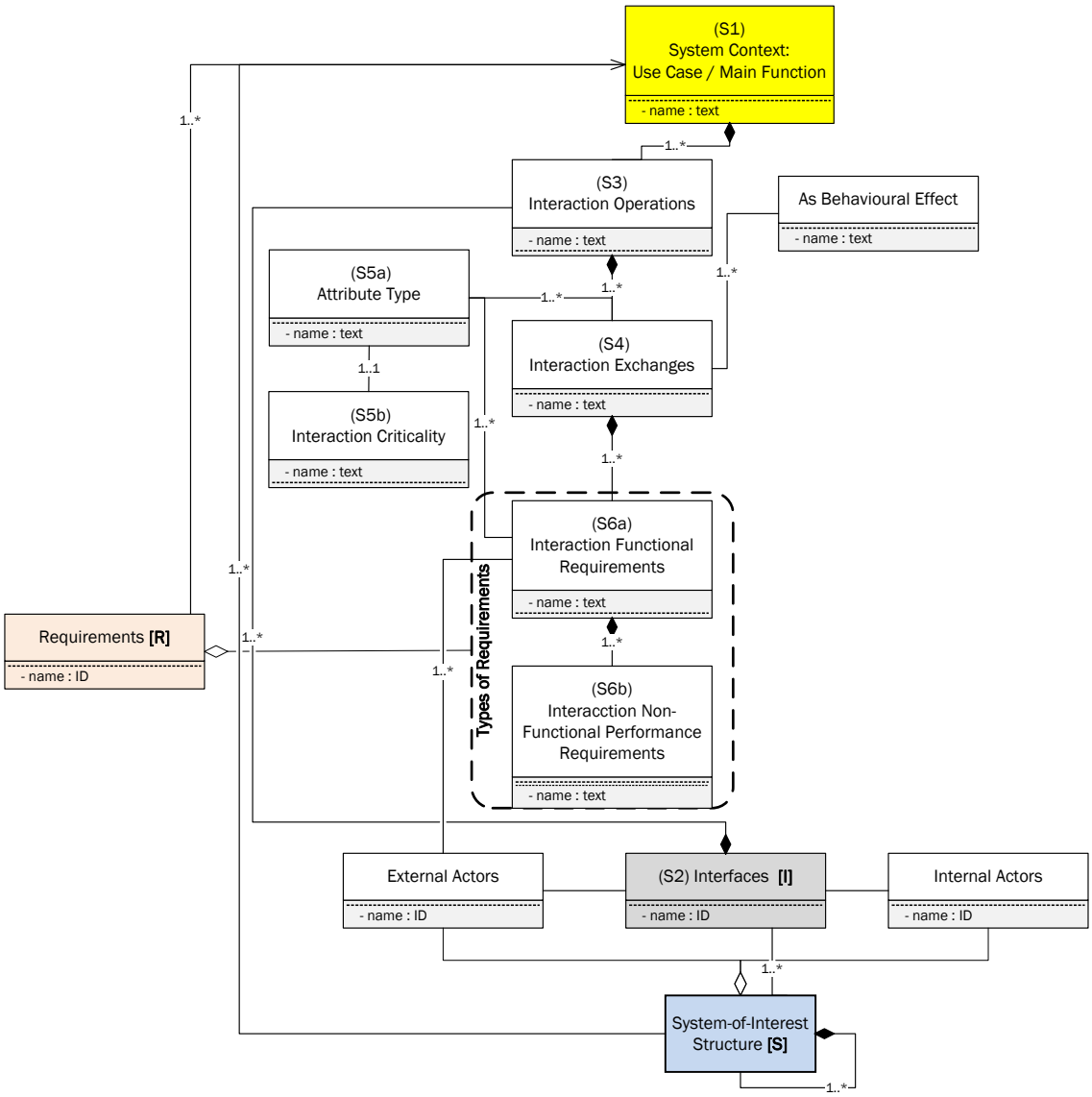


Figure 4.26 IAT model's viewpoints and their relations via UML model

The four blocks at the bottom of Figure 4.26 shows that a system-of-interest is used by external actors and in turn it is composed of internal actors. On one hand, it can be a *system* for its own super-system where it interacts with other environmental systems or external actors. On the other hand, a system-of-interest can be a *subsystem* of a system that interacts with other subsystems or internal actors of a system as well as with system's external actors. The external actors can be a user, designer, supporting enabling systems, and/or natural environment etc.

A system-of-interest requires interaction operations at its interfaces to achieve its high level use case or a function, as shown in the Figure 4.26 at the top. To achieve this high level use case, many interaction operations (either sequence based or non-sequence based) in different scenarios (main success or alternate flow) are possible among the interfaces of system-of-interest and its environmental actors. Each interaction scenario represents how a use case can be achieved, applied, used, or misused. The interaction operations can possess multiple interaction exchanges (both flows and form related) as behavioural effects that are in turn constrained and specified by interaction functional requirements. These interaction functional requirement identify what the system has to do in terms of inputs and outputs in association with its external interacting actors as well as for the fulfilment of high level use case. Along with it, each exchange effect on relevant type of attribute is specified and quantified or ranked using a five scale scheme as illustrated in Figure 4.26. The functional requirements are quantified by at least one or more (performance) requirements specifications. Both interaction functional and performance requirements represent types of requirements. An interface is assumed to be failed if it does not deliver the desired interaction functional and performance requirement when required as part of an overall high level engineered use case (goal) or main function.

To summarise, following structured information flow scheme is established (see Figure 4.27) to generate interface information from each of the steps explained earlier in a systematic manner;

- There can be many goals (both include & exclude) related to a system-of-interest;

- There can be one or many interfaces (both internal & external) to a system-of-interest (at black-box view i.e. external actors) and within it (at white-box view i.e. internal actors);
- There can be one or many interaction scenarios (both main success & exceptional flow) in an interface;
- There can be one or many interaction operations (both operations & structural related) in a scenario;
- There can be one or many interaction exchanges (minimum one & maximum four among E,M,I,P/S) in an interaction operation;
- There can be one or many interaction attributes affected by an exchange;
- There is only one interaction criticality associated with each attribute.
- There can be one or many interaction functional requirements associated with an interaction exchange type,
- There can be one or many non-functional (performance) requirements with an interaction functional requirement, and

This one-to-one and one-to-many relationships is already depicted in UML model.

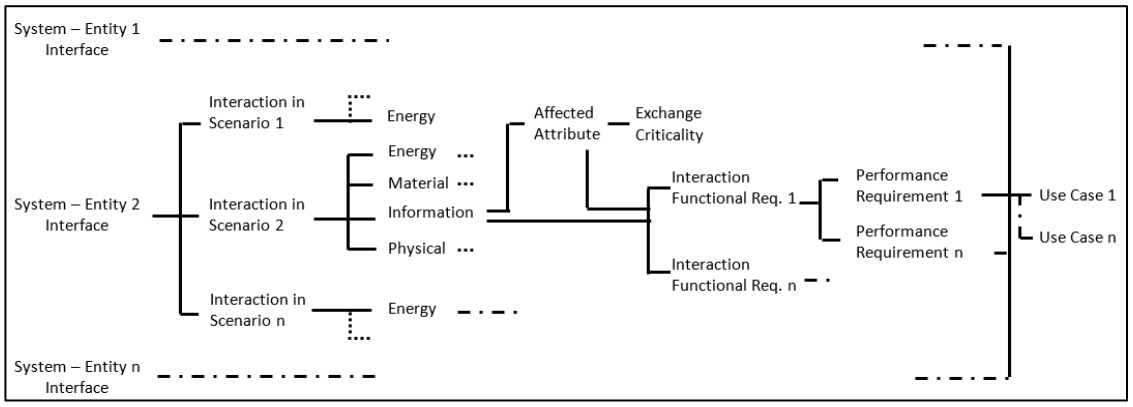


Figure 4.27 Information flow for interface analysis via IAT [C1 to C13]

4.6 Industrial applications

In previous sections, the relationships between interface modelling viewpoints', and the flexibility of IAT tool from working and thinking styles perspectives have been discussed. Also how it can support various design related activities. It is also shown how IAT tool can be applied in the system hierarchy at one level to capture the requirements with respect to stakeholders when its internal solutions are not known (black-box) and when subsystems known (i.e. white-

box). This is all shown on the basis of desktop case studies: ball point pen and electric pencil sharpener. These desktop case studies are purely physical systems that are easy to imagine and visualise the whole composition based on their subsystems.

However, in the real design world, system is not only composed of subsystems but also a set of multidisciplinary features (software and hardware related) that are delivered by the subsystems. New features design analysis even start before the system level. The features are defined and embedded into existing system architecture (e.g. vehicle) and these are often hard to visualise in contrast to existing or imaginable physical objects or systems. Therefore, it is important to check the applicability and integrity of IAT tool for relatively complex features as well as real world systems in the context of different levels and different nature of interfaces such as software and control aspects related. The IAT tool has been validated on following complex and real world automotive examples as illustrated in Table 4.14. The table also shows the vehicle hierarchy (i.e. system levels) and corresponding views of the case study where IAT was applied.

Table 4.14 The automotive applications used for IAT validation

Vehicle's system-of-systems (System-of-interests within the Vehicle hierarchy)	**IAT validation across views**	
	Black-Box	**White-box**
1.Feature Level – Appendix A (Junction view feature - software oriented)	X	
2.System Level – Appendix B (Regenerative braking system - electromechanically oriented)	X	X
3.Subsystem Level – Appendix C (Charger subsystem - hardware oriented)		X
4.Subsystem Level – Appendix D (Deployable active rear spoiler system – electromechanically)	X	

The analysis of these examples via IAT tool are available in Appendix A, B, C and D. In the next section, the key findings that were gained while implementing the IAT on real world automotive case studies are presented.

4.6.1 Key learnings

The proposed IAT tool is based on a tabular template and is derived from the review of current methods and tools used in the academic and industrial practice. The IAT tool supports the derivation of the system design requirements at early stages of design on the basis of integrated interface modelling viewpoints. It provides richer interface information in a systematic manner in contrast to the existing visual (graphical) and textual (tabular) tools that often work either in isolation for providing a specific view of a system or by-pass a modelling viewpoint or require comprehensive (linkage) integration among several tools for representing collective information. The validation process within an automotive industry highlighted the strengths and also weaknesses of IAT which are as follows;

- The key strength is that mechanical and software engineers used the IAT in slightly different manner specifically in the context of viewpoints ‘operation’ and ‘exchange’. Some engineers use one viewpoint i.e. exchange-based in more detail in contrast to operation viewpoint and vice versa while others used both interchangeably. In both cases, engineers managed to obtain a robust set of requirements thereby identifying requirements that were not discovered with previous existing requirements documentation processes. The existing requirements documentation processes in automotive industry are found to be use case and interaction operations based. Due to which it was found that requirements (both functional and performance related) were getting escaped which were discovered with the incorporation and consideration of exchange viewpoint with sharp focus on interface definition and thinking. Therefore, IAT proved its effectiveness across the real design environment. More detail discussion on gained insights and results on it are presented in detail in Chapter 6.
- The weakness of IAT is that it is time consuming due to many columns. It was also observed that many of the contents such as use case, and interaction operations were overlapping with existing documents in industry. The reason is obvious as IAT incorporates and integrates the viewpoints available in use case events-based and interaction exchanges-based templates.

4.7 Chapter summary

The overall aim of this chapter has been to introduce an IAT tool as a structured approach for deriving system-of-interest requirements in the systems hierarchy via interface based thinking.

The six-step based interface definition and modelling methodology is introduced and the steps are justified with the arguments on existing concepts around the ball point pen desktop case study. This helped in deriving a structured IAT tool systematically. The IAT is then tested on an electric pencil sharpener's black-box and white-box views in detail. Guidelines for using IAT in different design scenarios are also presented. It is shown that IAT has the ability to capture both desired and detrimental behavioural interactions and requirements; it can also be described a useful approach to improve the existing designs.

In the next chapter, it is seen how the IAT tool supports system architecture analysis activities; in terms of its integrated role for well-defined interfaces and in there the captured requirements in it that are allocated to solution independent functions to subsystems via Coupling Matrix (CM) framework.

5. Development of an Architecture Analysis Framework

5.1 Introduction

In this chapter, a framework to support system architecture analysis is proposed building on the views and viewpoints accumulated in the architecture cube-model 1 (Figure 3.3), which has revealed that existing modelling frameworks are insufficient in scope and procedure. The proposed framework integrates the *requirements derived via the IAT tool* (discussed in Chapter 4) at system's black-box with *transformative functions* that facilitate for solutions search, i.e. internal actors at white-box.

The reasoning leads to the development of the coupling matrix (CM) as a framework integrated with well-defined interfaces via IAT tool for system architecture analysis. The ball-point pen example is used with a view of justifying it based on existing concepts and gaps from literature. The CM framework supports the architecture analysis activities thereby coupling the system's black-box and white-box views for requirements and transformative functions allocation to internal actors (subsystems). The working principle of the CM framework is discussed via a desktop case study that yields to integrated architecture analysis methodology whose formalised model is represented via UML diagram.

5.2 The system architecture analysis key activities

In Chapter 4 (Section 4.2.2.9), it was discussed that the interaction functional requirements define the interactions between a system-of-interest and its external actors (black-box) as well as between the internal actors of a system-of-interest (white-box).

It was also discussed that solution independent functions identification is less emphasized in the environment centric and device centric functions concepts. The solution independent functions identification is one of the essential activities in system architecture analysis for the purpose of establishing a function structure (i.e. functional architecture). The function structure helps in exploring possible internal actors leading to physical architecture at white-box of a system. The function structure establishment can be referred to as an

intermediate step between system's black-box and white-box views. Thus, the approach proposed in this research supports the following two key system architecture analysis activities (as discussed in Figures 2.5, 2.6, and 2.15):

- allocation of high level captured functional and performance requirements (i.e. requirements view) from higher level to lower level solution independent decomposed functions (i.e. functional view);
- consideration of multiple/alternate candidate physical architectures based on solution independent functions, and their allocation to internal actors that can meet at least one functional requirement and may support many functions.

These two activities are now discussed in detail with reasoning on ball point pen example that results in the development of architecture analysis framework.

5.2.1 Identification of solution independent functions

As discussed in Chapter 4 (Section 4.2.2.8.1), in this thesis the solution independent functions are referred to *transformative functions* that are derived from in/out exchanges' (flows') transitions inside the system's boundary. The interaction functional requirements (device centric functions but solution dependent) are then captured among the chosen internal actors at white-box.

In existing state of art (see e.g. Crilly, 2012; Brown & Blessing, 2005), device centric functions discussion is limited to solution-dependent analysis with internal actors at white-box. It was argued in Chapter 4 that the given *device* centric function description by Brown & Blessing (2005) and Crilly (2012) is more like interaction functional requirement.

Although the examples given by Brown & Blessing and Crilly are device centric; the function descriptions are *internal actors specific* that describe the internal behaviour of pen device and not *transformation based or solution independent* functional descriptions related to the flow of operands/exchanges as recommended by Pahl et al. (2007) and Stone & Wood (2001) (see Table 2.6 in Chapter 2). This concept is illustrated in Figure 5.1 showing the high level function of pen device with its input and output main flow/exchange's states transition. Ink is the main essential exchange that has got two separate local

properties in association with paper (i.e. ink location 2, ink viscosity 1) and ink source (i.e. location 1, ink viscosity 1). Pen device is the mean to achieve the high level function. Here, author now applies black-box and white-box views' concepts also on functional descriptions at high and lower levels. For example, Figure 5.1 can be regarded as black-box function as it represents the high level function of a pen device which is often defined and practiced in similar way by researchers (Yildirim & Campean, 2014; Otto & Wood, 2001).

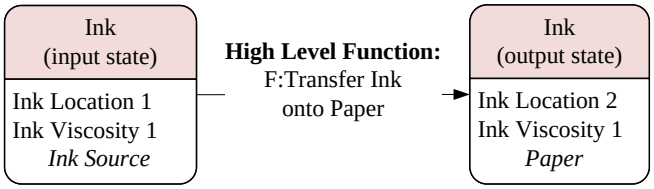


Figure 5.1 Pen device's high level function

The functional decomposition of the high level function 'Transfer Ink onto Paper' at black-box will lead to solution independent sub-functions that can be regarded equivalent to white-box view of high level function. Pahl et al (2007), and Stone & Wood (2001) describe the operations on flows/exchanges from initial state to final state by considering those as behavioural events (at white-box) or behavioural process as a whole (at black-box). All these behavioural events are represented with verb-noun or verb-object expressions revealing the operands and their state transformations. This sort of thinking associated with operands transformation results in meeting the physical laws fully as stated by Vermaas (2010);

> ***Behaviour is a physical phenomenon and meets the conservation laws as given by physics**. When **behaviour** can be taken **as an event**, the initial and the final states of affairs should contain equal amounts of energy, matter, charge, and so on. And **when behaviour** can be taken **as a process**, the input and output flows should again contain equal amounts of energy, matter, etc. For the behaviour of the lamp the incoming electricity and the outgoing light and radiation contain equal amounts of energy.*
>
> ***Behaviour** may be represented by, for instance, a **verb-noun expression** or by evolving **state variables of the device** and **the objects** that are interacting with it. If behaviour can be taken as a process, it can also be represented by operations of flows, as advanced by Pahl and Beitz & Stone and Wood.*
>
> ***Effects of behaviour of a device are now events and processes that are the result of behaviour of the device. These events or processes may consist in states of affairs consisting in the device itself and/or its properties (as in device-centric functions) as introduced by***

Chandrasekaran and Josephson [10], … (as in environment-centric functions), or in states of affairs that are combinations thereof.

The white-box operations are then used to explore and evaluate potential solutions/internal actors of the system that are referred to as white-box actors. Thus, it can be stated that the decomposed (sub) functions and internal actors appear at system's white-box view.

For the pen example, in order to come across solution independent functional decomposition, sub-functions need to be identified first that change the states or variables of the ink in Figure 5.1 from input to output (Section 2.4.5.2.4 and Figure 2.17). This is the stage where conservation laws should meet. Therefore, in this thesis such decomposed functions are referred to as *device centric* **transformative functions** that reveal functional architecture of a system. For example, 'Store Ink', 'Transfer Ink', and 'Disperse/Distribute Ink' would be device centric transformative functions that change the states of ink (i.e. exchange/flow) between its incoming state variables to its outgoing state variables via pen, thus meeting conservative laws, as shown in Figure 5.2 and Table 5.1 using the technique of state flow diagram (Figure 2.17).

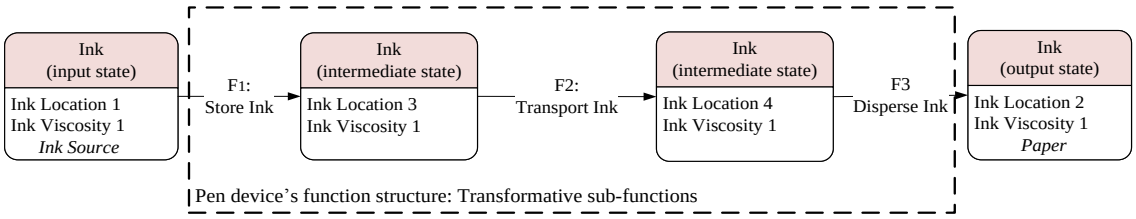


Figure 5.2 Device centric transformative functions

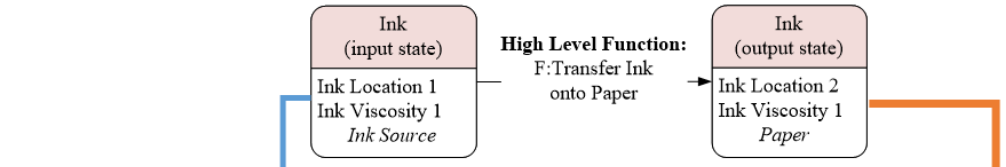
Table 5.1 Device centric transformative functions

	Exchange	Device Centric Statements
Operand Related	*M-Material* *(Ink)*	Operations related • ***Device Centric Transformative Functions:*** ***(****Solutions independent based)* *Store Ink, Transport ink, Disperse Ink*

Note that the state variables given by Brown & Blessing (2005) (Figure 4.2), appear to be device oriented and not main exchange (i.e. ink) related. Furthermore, such function descriptions, as in Figure 5.2, are not seen in the ball point pen examples given by Crilly (2012), and Brown & Blessing (2005)

during the discussion of device centric and environment centric function concepts.

On the other hand, the incoming and outgoing states of ink can be articulated through interaction but partially covering physics via environment centric **interaction functional requirements** such as pen device 'Imports Ink' from external object (ink source) and Delivers or 'Exports ink' to external object (paper) at black-box with same initial input and desired final output state variables with goal 'To Provide Information' as shown in Figure 5.3 via IAT.



IAT at Black-Box Actors

Step 2	Step 3	Step 4				Step 5	Step 6a	Step 1
Interface	Interaction Operation	Exchange & Type	Exchange Properties	From	To	Attribute Type & Effect Scale	(Environment Centric) Interaction Functional Requirements	Use Cases Goal 1: To provide information
Pen - Paper	Pen is in contact with paper	Ink (M)	Ink Location 2 Ink Viscosity 1	Pen	Paper	Performance (+2)	Export Ink	X
Pen - Ink Source	Pen is loaded with Ink	Ink (M)	Ink Location 1 Ink Viscosity 1	Ink Source	Pen	Performance (+2)	Import Ink	X

Figure 5.3 Environment centric Interaction functional requirements

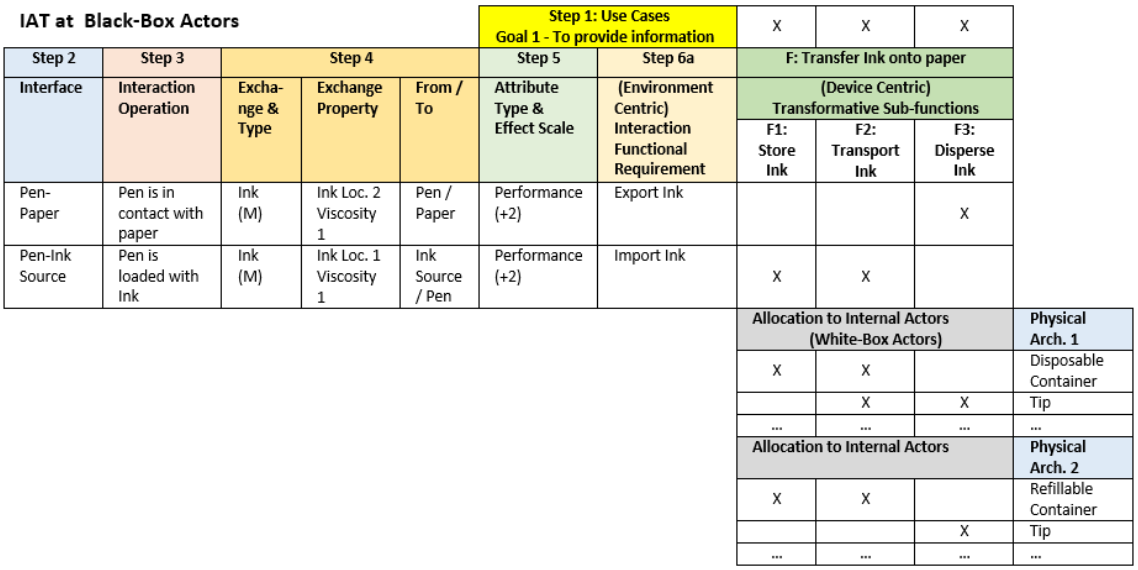
At black-box, in Figure 5.3, the *intermediate* states of ink cannot be thought for which white box analysis, in Figure 5.2, is recommended and is the common and well established practiced across engineering community. A chain of operations-on-exchanges/flows is then specified, called a *function chain* (Otto & Wood, 2001), for each black box related input flow (by prioritising first dominant flow & then the secondary flows), which transforms the main flow/exchange step-by-step into an output flow. Finally, these ordered function chains are aggregated into a single functional model of a product (Van Eck, 2010).

5.2.2 Consideration of multiple architectures and allocation of functions to subsystems

The functional architecture (or function structure) of a pen device, in Figure 5.2, is then used to explore potential internal actors. The key point to note in this research is that the explored internal actors would accomplish both environment centric **interaction functional requirements** at black-box (Figure 5.3) and

device centric **transformative functions** in Figure 5.2. After that, the device centric **interaction functional requirements** are specified in between the chosen internal actors. Therefore, it should be recognised that device-centric descriptions within the system boundary in turn can be categorised into following two concepts:

- Transformative functions - solution-independent based and;
- Interaction functional requirement- solution-dependent based.



IAT at Black-Box Actors					Step 1: Use Cases Goal 1 - To provide information		X	X	X
Step 2	Step 3	Step 4			Step 5	Step 6a	F: Transfer Ink onto paper		
Interface	Interaction Operation	Excha-nge & Type	Exchange Property	From / To	Attribute Type & Effect Scale	(Environment Centric) Interaction Functional Requirement	(Device Centric) Transformative Sub-functions		
							F1: Store Ink	F2: Transport Ink	F3: Disperse Ink
Pen-Paper	Pen is in contact with paper	Ink (M)	Ink Loc. 2 Viscosity 1	Pen / Paper	Performance (+2)	Export Ink			X
Pen-Ink Source	Pen is loaded with Ink	Ink (M)	Ink Loc. 1 Viscosity 1	Ink Source / Pen	Performance (+2)	Import Ink	X	X	

Allocation to Internal Actors (White-Box Actors)			Physical Arch. 1
X	X		Disposable Container
	X	X	Tip
...	...	...	...
Allocation to Internal Actors			**Physical Arch. 2**
X	X		Refillable Container
		X	Tip
...	...	...	...

Figure 5.4 Device centric descriptions and multiple architectures creation

Figure 5.4 shows how device centric transformative functions can be connected with environment centric functional requirements (at black-box) via a matrix and also their allocation to multiple architectures. For example, in Figure 5.4, there are two possible physical architectures for a pen device. Physical architecture 1 involves Disposable Ink Reservoir/Container actor whereas architecture 2 involves Refillable Container actor whilst the other actor i.e. Tip remains same in both architectures. Both physical architectures at white-box can deliver the desired functionalities. For example, Ink Container or Reservoir would serve ‘Store Ink & Transport Ink’ (both transformative but device centric at white-box) and ‘Import Ink’ from Ink Source (interaction but environment centric at black-box) whilst Tip would serve ‘Transport Ink & Disperse Ink’ (both transformative but device centric) and ‘Export Ink’ to paper (environment centric interaction) as illustrated in Figure 5.4. It should be noted here that a single function can be served by two internal actors (recall Section 2.4.7); in this case ‘Transport Ink’ is

allocated to two internal actors of a pen i.e. both Ink Container and Tip. Thus such matrix-based approach would also support both modular and integral types of architectures (as highlighted in Section 3.6.2.1 in Chapter 3 as one of the key requirements on architecture analysis framework).

It should also be noted that transformative functions would help to achieve the end goal which is 'To Provide Information' in Figure 5.3 in IAT that can also be mapped in matrix-based approach as shown in Figure 5.4. Based on design criteria (not a scope of this thesis), physical architecture no. 1 can be chosen for further specifying interaction functional requirements (in step 6a) in between its internal actors in IAT along with relevant mapped transformative functions (step 1) in it as shown in Figure 5.5.

IAT at White-Box Actors

Step 2	Step 3	Step 4				Step 5	Step 6a	Step 1
Interface	**Interaction Operation**	**Exchange**	**Exchange Properties**	**From**	**To**	**Attribute Type & Effect Scale**	**(Environment Centric) Interaction Functional Requirements**	**Transformative Sub-function(s)**
								Transport Ink
Disposable Container - Tip	Allow ink flow between container and tip	Ink (M)	Ink Location 4 Viscosity 1	Container	Tip	Performance (+2)	Export Ink (From Container Perspective)	X
							Import Ink (From Tip Perspective)	X

Figure 5.5 Interaction functional requirement via IAT between internal actors

The matrix-based approach in Figure 5.4 is named as coupling matrix (CM) which is now discussed in detail in the next section.

5.3 Overview of CM framework

The views and viewpoints of architecture cube-model 1 (Figure 3.3) are arranged into developed CM, illustrated graphically in Figure 5.6. Figure 5.6 also shows the key associated viewpoints of IAT, integrated with CM framework.

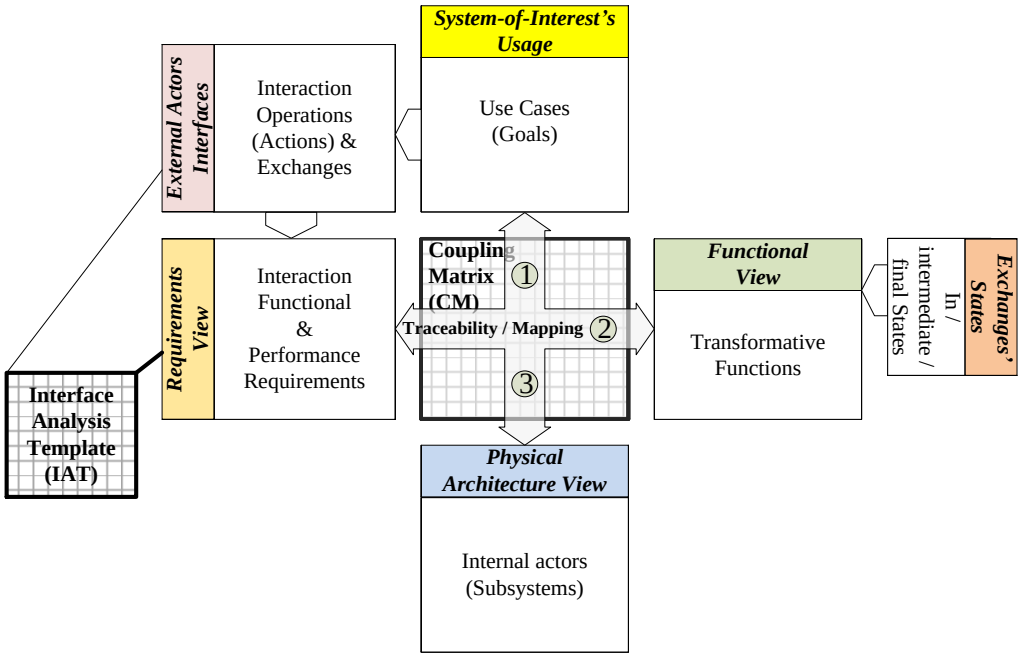


Figure 5.6 Overview of CM framework in conjunction with IAT

Figure 5.6 shows that the system interaction requirements at black-box are captured via IAT (as discussed in Chapter 4) through use case, interaction operations, and exchanges viewpoints associated with external actors of the system. The use cases of the system are linked to system's transformative functions in sub-matrix 1 of CM framework. The requirements captured at black-box via IAT are allocated to decomposed transformative functions of a system in sub-matrix 2. The coupled relationships in matrix 2 are then allocated to internal actors/subsystems in sub-matrix 3. These three matrices are together integrated into a CM framework. The working of CM framework in conjunction with integrated IAT will now be discussed via desktop example in next section.

5.3.1 Desktop example: Coffee vending machine

Coffee vending machine involves handling of many main exchanges/flows (such as coffee beans and water) and there are many transformative functions associated with it. Also this case study has been previously analysed in literature by many researchers (e.g. Eisenbart, 2014). Therefore, the effectiveness and efficiency of the proposed approach can also be benchmarked with this case study. Note that the CM framework in this chapter is validated with coffee vending machine at one level of decomposition.

5.3.1.1 Requirements analysis: Use cases vs interaction requirements via IAT

The coffee vending machine involves a variety of use cases as illustrated in the use case diagram in Figure 5.7. Interaction requirements can be extracted via IAT either by considering a single use case or multiple use cases at a time (as discussed in Section 4.3.1.3). The interface modelling methodology include (a) use case diagram (b) system context diagram and then (c) IAT tool.

A use case diagram, in Figure 5.7, illustrates four use cases of coffee vending machine associated with its external actors. This diagram also shows the dependency between use cases e.g. ‘prepare a cup of coffee’ involves ‘preparing a hot tea water’ which is represented by an include relationship. The extend relationship can also be used to specify additional changes that can be made to the base use case.

Use cases:

- prepare a cup of coffee,
- Prepare a cup of green tea,
- prepare a cup of cappuccino, and
- prepare hot tea water.

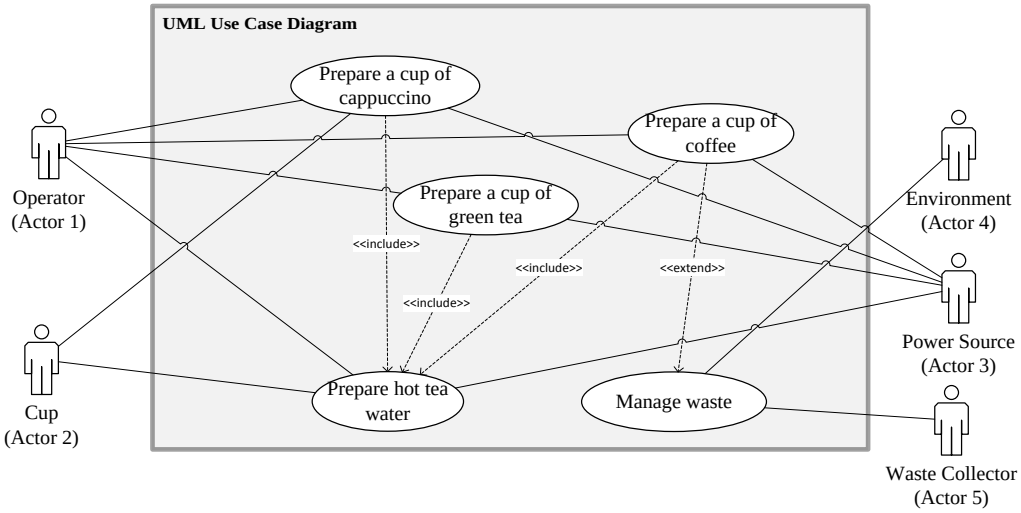


Figure 5.7 Use cases for coffee vending machine

Alternately, at this level of abstraction, use case diagram can be transformed into system context diagram illustrated in Figure 5.8, where a system’s working

boundary with its surrounding (external) actors can be visualised as an interfaces.

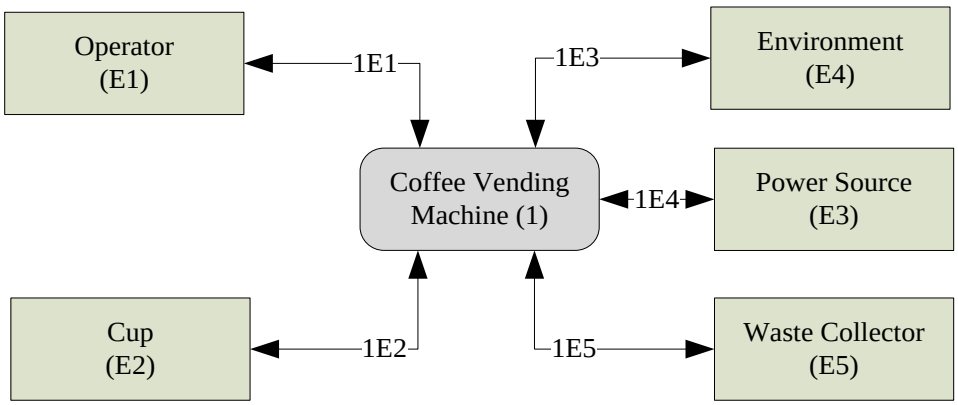


Figure 5.8 System context diagram for coffee vending machine

In the next step, the interactions between the coffee vending machine system and external actors are analysed via the IAT. This can be done either two ways: sequential or non-sequential thinking (as discussed in Section 4.4). Here, the non-sequential approach was employed for the coffee vending machine analysis to derive its relevant functional and non-functional performance requirements. For example, the interaction functional requirements in the use case ‘prepare a cup of coffee’ can be common across other use cases of coffee vending machine, however non-functional performance requirements may or may not vary from use case to use case. For instance, 1E1-1 interaction functional requirement ‘import water’ (or in traditional shall language, ‘system shall import/receive water’) is also involved in other use cases: ‘prepare hot tea water’ and ‘prepare a cup of cappuccino’ and for which non-functional requirements are same as shown in Figure 5.9. The non-functional performance requirements in the use cases ‘prepare a cup of coffee’ and ‘prepare a cup of green tea’ would also remain same for the 1E2-1 interaction functional requirement ‘Deliver hot water’; e.g. the amount of water 120 millilitres and the temperature around 363 to 373 Kelvin. However, these quantities (non-functional performance related) in interface 1E2-1 would be different for the other use case ‘prepare a cup of cappuccino’ as shown in Figure 5.9 in IAT. Thus IAT supports in capturing both functional and non-functional (performance) requirements across multiple use case.

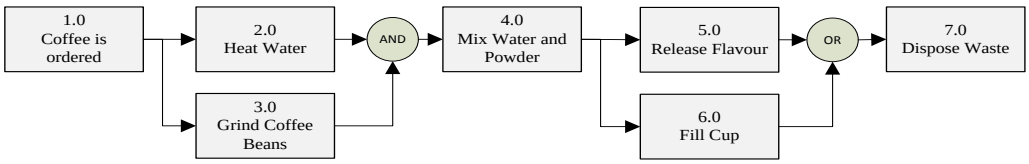
S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				S1: System Use Cases		
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13		
Interf.	Interac. Step	Operations Description	Exc. Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Functional Requirement Verb	Object/Non-Object Input	Output	Nonfunctional Requirement (Performance Related)	UC1: Prepare a cup of coffee	UC 2: Prepare hot tea water	UC3: Prepare a cup of cappuccino
Operator (Op) - Coffee Vending Machine (CVM) Interface	1E1-1	Operator puts coffee preparation material into coffee vending system	M	Coffee Beans	Beans size	Op	CVM	A3: Accommodation & Usage	2	Import	Coffee Beans		Coffee beans intake capacity (X < mm ^ 3 < Y)	X	X	X
			M	Normal Water	Water temperature Specific heat capacity	Op	CVM	A3: Accommodation & Usage	2	Import	Normal water		Water temperature (287 < Kelvin<297) Water intake capacity (X < Lit <Y)	X	X	X
	1E1-3	Operator activates the coffee vending system by opting a coffee type	I	ON/OFF commands	On/Off Signals	OP	CVM	A9: HMI & Audio Visual Performance	2	Accept	ON/Off commands		On/Off commands signals (X < mA < Y) Response time (X < sec < Y)	X	X	X
	1E1-4	Coffee system provides warm brew coffee	M	Brew coffee	Coffee composition Coffee color	CVM	OP	A3: Accommodation & Usage	2	Deliver		Brew coffee	Coffee temperature (X < degress Celcius < Y)	X	X	X
Cup - CVM Interface	1E2-1	Coffee system generates warm water according to opted coffee type	M	Hot water	Water temperature Specific heat capacity Heat of vapourisation	CVM	CUP	A7: Performance		Deliver		Hot water	Water temperature (363 < degrees Celcius <373) Quantity of water (120 +/- 0.5 ml) Quantity of water (135+/- 0.5 ml) Time duration to warm water (X+/-y secs) Water flow rate (X < lit/sec < Y)	X X	X	X X X

Figure 5.9 Coffee vending machine's use cases and interaction requirements

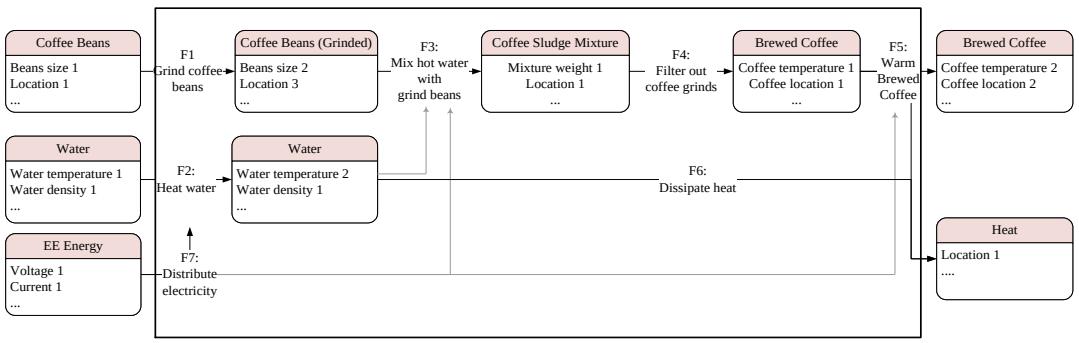
The next section shows a functional view of system; the decomposed transformative functions that are involved in the accomplishment of use case(s) or high level function.

5.3.1.2 Functional analysis: Transformative functions

The transformative functions within a use case of ‘prepare a cup of coffee’ are now explored. Note that the transformative functions of the system in a particular use case can also be analysed and represented through a number of functional modelling tools (Section 2.4.5) such as functional block diagram, functional flow diagram, function tree or/and mind mapping other than states based representation (as discussed in Section 5.2.1). These functional modelling tools help in providing the *function structure* or functional architecture of a system. The transformative functions of the coffee vending machine are shown in Figure 5.10 a&b from F1 till F7 via two different techniques: functional flow block diagram (without states based thinking) and system state flow diagram (with states based thinking). These tools represent the order in which the functions are to be carried out with each other. For example, ‘F1: grind coffee beans’ and ‘F2: heat water’ may work in parallel as shown in Figure 5.10 a&b. It should also be noted that the number of the transformative functions or the function structure of a system may or may not vary from one tool to another tool due to following two reasons: either because of technical skill of the analyst or because of the heuristics or the structured technique of a functional modelling tool as evident from Figures 5.10 a&b.



(a) Functional flow block diagram



(b) System state flow diagram

Figure 5.10 Transformative functions of coffee vending machine

The next step for architecture analysis is to check and link the transformative functions that contribute to the other use cases of the coffee vending machine.

5.3.1.3 Mapping transformative functions vs use cases via CM

The function structure derived for a use case may involve transformative functions belonging to other use cases that can be mapped via matrix 1 of CM as illustrated graphically in Figure 5.11 (with or without states based thinking is adopted). For example, in Figure 5.12, functions ‘F2: heat water’ and ‘F6: distribute electricity’ of a use case ‘prepare a cup of coffee’ also contribute to other use cases such as ‘prepare hot water’, and ‘prepare a cup of green tea’ and hence mapped in matrix 1 of CM.

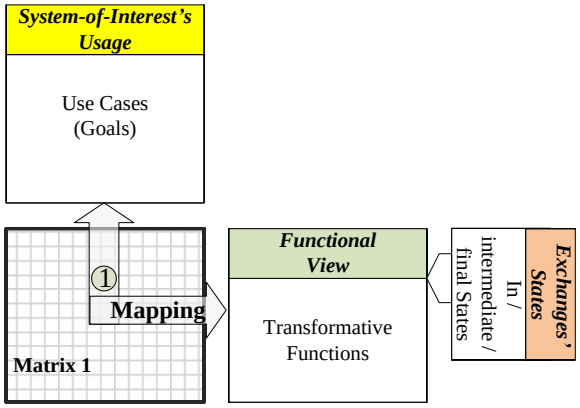


Figure 5.11 Matrix 1: use cases and transformative functions

System Use Cases								
	UC 4: Prepare a cup of green tea		X				X	X
	UC 3: Prepare a cup of cappuccino	X	X	X	X	1	X	X
	UC 2: Prepare hot tea water		X				X	X
	UC 1: Prepare a cup of coffee	X	X	X	X	X	X	X
		Transformative Functions						
		F1: Grind Coffee Beans	F2: Heat Water	F3: Mix Water with Grind Beans	F4: Filter out Coffee Grinds	F5: Warm Brewed Coffee	F6: Distribute Electricity	F7: Dissipate Heat

Figure 5.12 Coffee vending machine’s use cases and transformative functions

5.3.1.4 Mapping transformative functions vs interaction requirements

In the next step, the transformative functions that would deliver the interaction requirements (in IAT) are mapped/coupled in matrix 2 of CM as illustrated graphically in Figure 5.13. Two functions can be responsible for accomplishing a single functional requirement and it is also possible that a single function contributes to more than one functional requirement. For example, in Figure

5.14, both functions ‘F2: heat water’ and ‘F3: mix water with grind beans’ within a use case ‘prepare a cup of coffee’ contribute to a single interaction functional requirement 1E1-1 ‘import water’ as well as share same non-functional performance aspect i.e. ‘temperature of water’. Therefore, the proposed framework has the ability to manage such complex information thereby coupling (mapping) the transformative functions of the system to its relevant interaction functional and performance requirements associated with different types of exchanges.

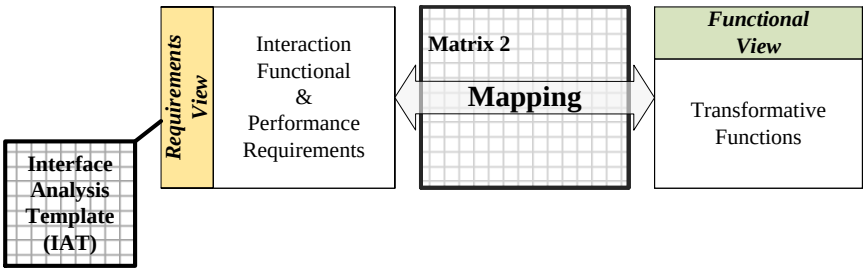


Figure 5.13 Matrix 2: transformative functions and interaction requirements

S2: Interface	S3: Specifications of Interaction Operations		S4:Specification of Exchanges		S5: Exchange Effect / Impact		S6: System of Interest Requirements				Transformative Functions		
C1	C2	C3	C4	C5	C9	C10	C11			C12			
Interf.	Interac. Step	Operations Description	Exc. Type	Exchange Description	Attribute Type	Scale	Functional Requirement: Verb	Object: Input	Object: Output	Nonfunctional Requirement (Performance Related)	F1: Grind Coffee Beans	F2: Heat Water	F3: Mix Water with Grind
Operator (Op) - CVM Interface	1E1-1	Operator puts coffee preparation material into coffee vending system	M	Coffee Beans	A3: Accommodation & Usage	2	Import	Coffee Beans		Coffee beans intake capacity (X < mm ^ 3 < Y)	x		x
			M	Normal Water	A3: Accommodation & Usage	2	Import	Natural water		Water temperature (287 < Kelvin <297)	x	x	x
										Water intake capacity (X < Lit <Y)	x	x	x
	1E1-3	Operator activates the coffee vending system by opting a coffee type	I	ON/OFF commands	A9: HMI & Audio Visual Performance	2	Accept	ON/Off commands		On/Off commands signals (X < mA < Y) Response time (X < sec < Y)		2	
	1E1-4	Coffee system provides warm brew coffee	M	Brew coffee	A3: Accommodation & Usage	2	Deliver		Brew coffee	Coffee temperature (X < Kelvin < Y) Quality of coffee (X +/- y ml)			x
Cup - CVM Interface	1E2-1	Coffee system generates warm water according to opted coffee type	M	Hot water	A7: Performance		Deliver		Hot water	Water temperature (363 < degrees Celcius <373)		x	x
										Quantity of water (120 +/- 0.5 ml)		x	x
										Quantity of water (135 +/- 0.5 ml)		x	x
										Time duration to warm water (X+/-y secs)		x	x
										Water flow rate (X < lit/sec < Y)		x	x
	...	...	...	...	...	...	...	...	...	...	...	...	...
Power Sources (PS) - CVM Interface	1E3-1	Electric mains supply electric energy to coffee vending system	E	Electric energy	A11: Energy Management		Import	Electricity		Voltage (X +/- y volts)	x	x	x
										Current (X +/- y mA)	x	x	x

Figure 5.14 Coffee vending machine’s functions and interaction requirements

5.3.1.5 Coupled transformative functions vs interaction requirements allocation to subsystems

Once, it is identified which transformative functions contribute to which interaction requirements in a use case or use cases of a system, then in the next step, multiple physical architectures are explored. The mapped transformative functions vs interaction requirements in matrix 2 (Figure 5.14) are assigned to subsystems/internal actors at white-box via matrix 3 of CM as

illustrated graphically in Figure 5.15. The internal actors/subsystems in a physical architecture can be mechanical, electrical or/and software systems.

Note that there can be many physical architectures, as an example in Figure 5.16, that may support the functional structure (or functional architecture) developed in Figure 5.10 but at the same time it is important to check that those also satisfy interaction requirements (as mapped in Figure 5.14). For a best possible physical architecture selection, there can be set of trade-off criteria (such as other non-functional aspects such as reliability, cost, weight, etc.) which design team can develop (and is not a main theme of this research). Note that a single function can be achieved by more than one internal actor and vice versa. For example, in Figure 5.17 a single subsystem/internal actor 'heating subsystem' serves functions 'F2: heat water' and 'F3: mix water with grind beans' whereas 'F2: heat water' is assigned to 'heating system' as well as 'control system'. The coffee vending machine in Figure 5.17 represents an example of integral architecture type.

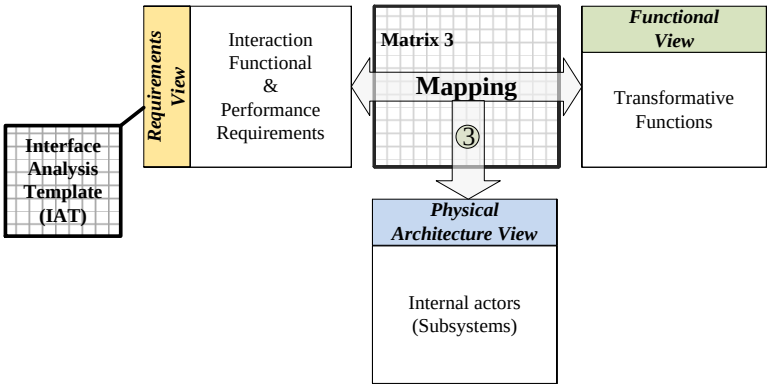


Figure 5.15 Matrix 3: transformative functions vs interaction requirements set allocation to subsystems

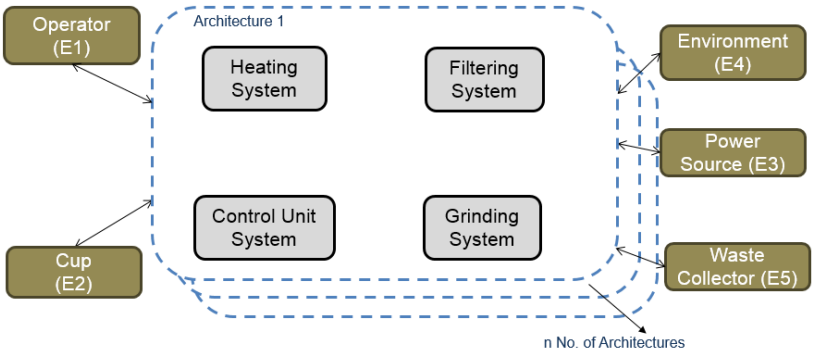


Figure 5.16 System boundary diagram without relationships

Figure 5.17 Transformative functions and interaction requirements allocated to technical subsystems

The CM framework at this stage provides internal actor/subsystem's requirements e.g. 'heating subsystem' will serve following set of system's transformative functions, and interaction functional and performance requirements:

A. Transformative functions;

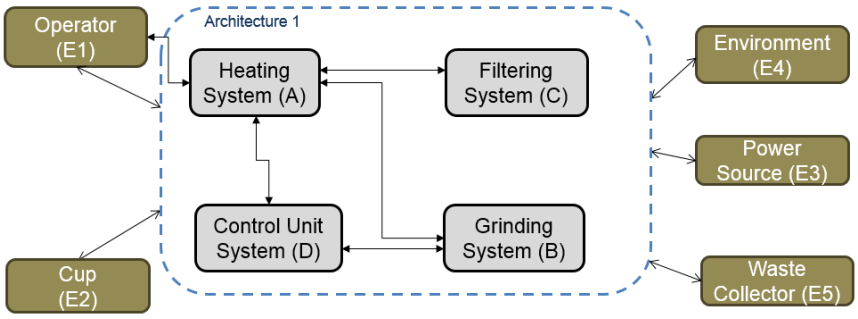
- F1: Heat Water
- F3: Mix Water with Grind Beans
- F5: Warm Brewed Coffee &
- F7: Dissipate Heat

B. Requirements flow down from system's black-box to system's white-box:

- **Interaction functional requirements;**
  - Import electricity
  - Reduce heat dissipation
  - Reduce noise
  - Import water
  - Import coffee beans
  - Accept on/off command
  - …
- **Interaction non-functional performance requirements**
  - On/off command signals
  - Intake water temperature
  - Output water temperature
  - …

5.3.1.6 Detail physical architecture view: Interaction requirements

The chosen architecture based on trade-off criteria is then detailed further via system boundary diagram as shown in Figure 5.18 for revealing abstract interaction information among subsystems, and again IAT is used for capturing detail information at white-box as shown in Figure 5.19.



Arrows & Lines Key: External interaction requirements of system cascaded to subsystems (see Fig. 5.17)

Internal to internal & internal to external interactions

Figure 5.18 System boundary diagram including relationships

S2: Interface	S3: Specifications of Interaction Scenarios		S4:Specification of Exchanges					S5: Exchange Effect / Impact		S6: System of Interest Requirements				System Transformative Functions			
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13			
Interf.	Interac. Step	Interaction Description	Exc. Type	Exchange Description	Exchange Properties	From	To	Attribute Type	Scale	Functional Requirement Verb	Object/Non-Object Input	Output	Nonfunctional Requirement (Performance Related)	F1: Grind Coffee Beans	F2: Heat Water	F3:Mix Water	F4: ...
Control Unit System (CUS) - Grinding System (GS) Interface	DB-1 - CUS sided view	Control unit activates the grinding system	I	Control signal	Signal current	CUS	GS	A11: Energy Management	1	Transmit		Control signal	Electric current for signal (X< mA <Y) Response time (X<msec<Y)	x			
	DB-1 - GS sided view	Grinding system requires control unit signal to start	I	Control signal	Signal current	CUS	GS	A11: Energy Management	1	Receive	Control signal		Electric current for signal (X< mA <Y) Response time (X<msec<Y)	x			
	...	...	...	...	...	...	...	...	...	...	...	...	...				
Heating System (HS) - Control Unit System (CUS) Interface	DA-1 - CUS sided view	Control unit system performance affects due to thermal energy exchange	E	Thermal energy	Heat energy	HS	CUS	A4: Thermal	-2	Reduce / Prevent	Thermal energy exchange		Heat dissipation rate (X < J/s < Y) Thermal isolation standard - XXX		x		
	DA-1 - HS sided view	Heating system dissipates thermal energy to CUS	E	Thermal energy	Heat energy	HS	CUS	A4: Thermal	-2	Reduce / Prevent		Thermal energy exchange	Heat dissipation rate (X < J/s < Y) Thermal isolation standard - XXX		x		
	...	...	...	...	...	...	...	...	...	...	...	...	...				
Operator (Op) - Heating System (HS) Interface	AE1-1 HS sided view	Heating system may transfer heat to operator	E	Thermal energy	Heat	HS	Op	A4: Thermal		Isolate		Heat dissipation	Heat isolation standard (ET-XXX)		x		
	...	...	...	...	...	...	...	...	...	...	...	...	...				
	_	_	_	_	_	_	_	_	_	_	_	_	_				

Figure 5.19 Coffee vending machine subsystems' interaction requirements via IAT and linkage with transformative functions

Note as now internal actors are known, at this stage, new interaction requirements would appear between the interfaces of external and internal actors. This is carried out using the standard IAT (as discussed in Section 4.3.2). However, the question arises that should new discovered interaction requirements at system's white-box be updated at black-box too or not? Should we capture and keep that information at the internal actors' level (i.e. white-box) or also put them at one level up with external actors (i.e. black-box)?

The answer to this is that it should be done at both levels and this can be iterative. Why is it so? The reasons to this is explained below with few examples.

As shown in Figure 5.19, It is now known that 'heating subsystem' will be used to heat the water, it might dissipate some heat to environment, user and also

transfer heat to water. These were not obvious at one level up or are often hard to discover at one level up (black-box) when a new system is designed. If we capture such sort of interface requirements for an internal actor (looking at internal to external interactions e.g. heating system with water, heating system with environment and heating system with user) then we may be capturing a complete set of interactions (both derisible and undesirable) that can help us to shortlist the best possible solution out of number of solutions for a same actor (i.e. heating subsystem) whereas if we don't put such requirements associated with exchanges such as heat dissipation, mechanical vibrations, and noise etc. at black-box level for a whole system (i.e. coffee vending machine) then design team may never be in position to judge how this whole system behaved in the past with respect to external actors and also what went wrong and why. The user experiences such sort of aforementioned effects from a system as a whole and she/he is usually not aware internal actors/subsystems that cause such effects.

Furthermore, if designer also updates the black-box IAT of coffee vending machine with such white-box level knowledge, it will also help to engineers at black-box level to shortlist and compare best possible solution out of many coffee vending systems existing today upfront in the design process. It will also help in understanding that how this system can behave (keeping in view of both desirable and undesirable interactions) if it has to become a part of a super system.

Now how can we document and place these requirements at both levels? First, at black-box level, as this rule has been established that a designed system which is a part of environment that has to fit with or got other external actors (e.g. user, AC source, environment etc.), its requirements should be captured in solution independent manner. Therefore, requirements related to heat dissipation, noise, vibrations may not be possible to capture first as evident from Figures 5.9, 5.14 & 5.17. However, these new requirements appear after knowing the internal actors or when form comes into existence and thus iteratively can be placed at black-box IAT without specifying the internal actors in textual descriptions as shown in Figure 5.20 in CM labelled as 1E4-1, 1E4-2, and 1E4-3 with environment interface. In future design, this can prompt to

design team what were the undesired interactions with this technology or how it behaved without looking at the internal structure of the system.

At white-box level it is also important to keep these requirements, because for next critical system analysis, those requirements would be mapped against that next critical system's transformative functions. For example, in this case, heating subsystem can become a next critical level system for analysis and it is important to capture its interactions not only with super system's external actors (i.e. coffee vending machine) such as user and environment (i.e. internal to external) but also with its technical neighbouring enabling systems at the same level such as Control subsystem, and Grinding subsystem etc. which also become external to heating subsystem as illustrated in Figure 5.19.

5.3.1.7 A synthesised interpretation of system architecture via CM framework

Figure 5.20 represents the complete integrated framework of CM based on three sub-matrices in conjunction with IAT which were discussed in previous sections. The use case viewpoint requires re-configuration from IAT tool (Figure 5.9) to CM framework (Figure 5.20) in matrix 1 when mapping between transformative functions and use cases is needed. The CM represents and unites *requirements* (i.e. both interaction function and performance related coming from external actors' interfaces) via IAT to *functional architecture (i.e. transformative functions)* via matrix 2 and also their allocation to *physical architecture (i.e. internal actors)* via matrix 3.

The CM framework supports in analysing the architecture analysis from abstract to detail information among the different integrated views and viewpoints in a flexible manner without changing the views positions. For example, if a designer wants to study the functional architecture of a coffee vending machine with its interaction requirements to external actors then only matrix 2 can be visualised and the other views can be kept hidden (i.e. matrices 1 and 3). Similarly, if a designer wants to study the functional architecture view with the internal actors/subsystems (physical architecture view) of the coffee vending machine without looking into the details of its requirements (i.e. matrix 2), then matrix 3

can only be visualised. Also if a designer wants to study the use cases (goals) of the system with its functional architecture view at an abstract level without going into concrete details behind the requirements and physical views then matrix 1 can only be visualised by keeping the other two matrices hidden (i.e. matrix 2 and 3). Thus, the CM framework in conjunction with IAT supports system architecture analysis in both effective manner along with flexibility of switching between different views.

Figure 5.20 System architecture analysis of coffee vending machine

Note that order of performing coupling relationships between matrices can vary i.e. different entry points. For example, in contrast to sections 5.3.1.3 and 5.3.1.4, the design team can start other way round i.e. mapping transformative functions with interaction requirements first via matrix 2 for a single use case ‘prepare a cup of coffee’ and later mapping its transformative functions to other use cases of the system via matrix 1. Thus, the CM framework also allows different entry points with the centralised view of transformative functions.

The CM framework’s validation in conjunction with well-defined interfaces via IAT along with other functional modelling tools on the desktop case studies (i.e. ball point pen and coffee vending machine) leads to an integrated architecture analysis framework comprising of following activities as discussed from sections 5.3.1.1 to 5.3.1.6.

(1) Identification of system context with its interaction (functional and performance) requirements from external actors (via use case or/and context diagrams, and IAT tools);
(2) Identification of system’s transformative (decomposed) functions (via any functional modelling tool);
(3) Allocation of interaction requirements to transformative functions (via CM framework);
(4) Identification of system’s physical architectures with internal actors/subsystems (design candidates);
(5) Allocation of interaction requirements vs transformative functions to subsystems (via CM framework);
(6) Analyse interfaces between the internal actors of a chosen architecture (system boundary diagram, and IAT tools);
(7) Repeat cycle for next system-of-interest at lower level.

The framework should recursively be applied across at each level of systems hierarchy.

5.4 Formalised model of CM framework

The UML diagram is used to represent the formalised model of CM framework in conjunction with integrated IAT. It is observed that many coupling relationships (one-to-one and one-to-many) between various viewpoints of

different modelling views are required as summarised in Figure 5.21 in order to create and manage system architecture in a systematic manner.

The intended or perceivable *behaviour* of a system under consideration is generally analysed collectively via four key views: *requirement*, *function*, *structural* and *interface* modelling. The entities in the CM framework and their relations are defined in accordance with the notions of key modelling views (Figure 2.1). In Chapter 4, the *structural*, *requirement,* and *interface* related viewpoints were presented in the formalised model of IAT via UML diagram. In Figure 5.21, additional representative viewpoints related to *functional view* (such as transformative function and states) for creating system architecture via CM framework are incorporated and integrated in the same formalised model of interface modelling methodology (Figure 4.26).

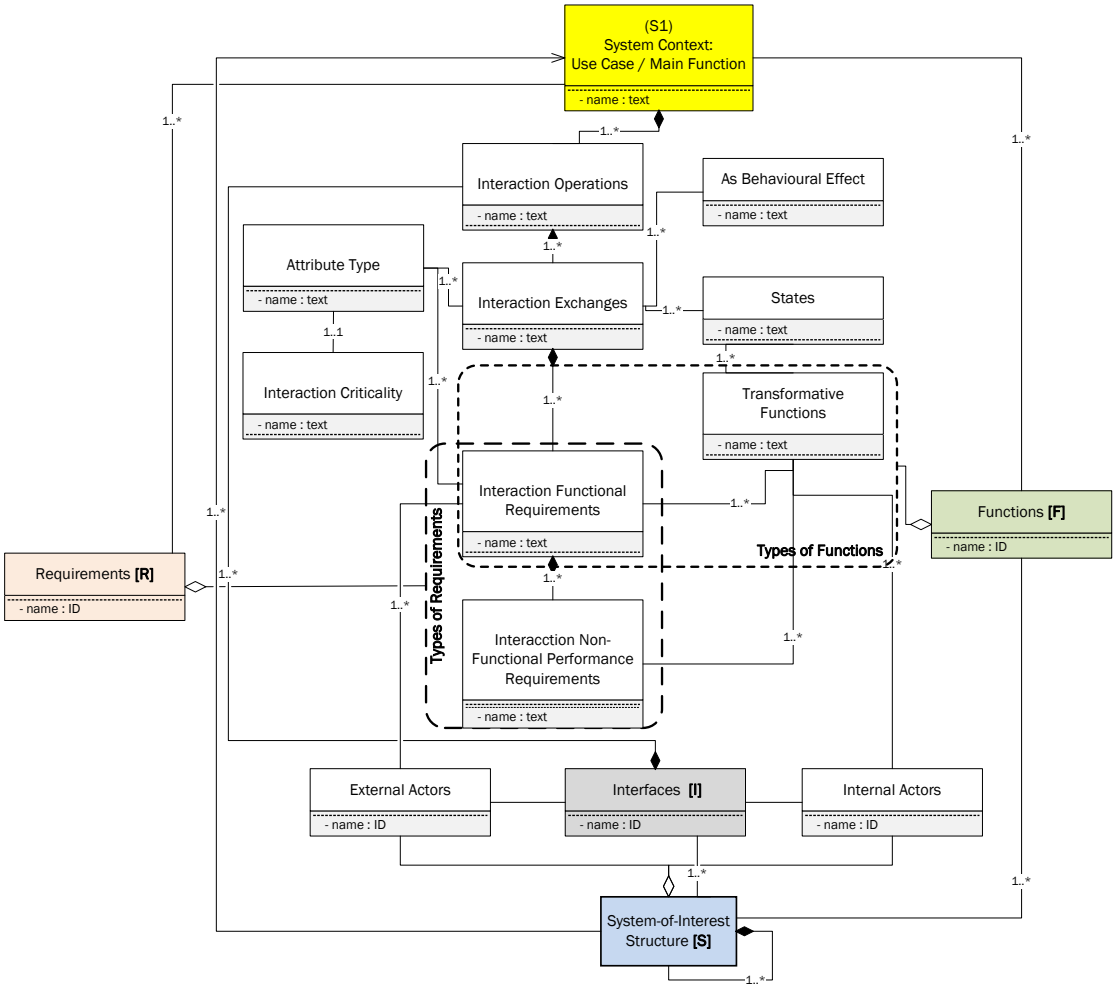


Figure 5.21 UML class diagram for CM modelling framework for system architecture development

The types associated with each view e.g. for requirement (i.e. functional and non-functional performance related) and for function (i.e. transformative function and interaction functional requirement) are also highlighted in Figure 5.21.

The requirement, function, structure and interface key views in turn contain following sub-viewpoints aggregated together via one-to-one and one-to-many relationships: *use cases*, *interaction operations (as events)*, *exchanges* (both *flows/operands* and *form* related), their *states* & *properties*, *attributes*, *criticality*, *functional & non-functional requirements*, *transformative functions*, and *internal & external actors*.

The *behaviour view* is the result of combination of the four key views (i.e. interface, function, structure, and requirement). With regard to the modelling viewpoints in the Figures 5.21, the intended but an abstract behaviour may be regarded through the consideration of different use cases, the operations (events), and the exchanges within an interaction that may occur between a system and external actors at its interfaces in Figure 5.9 that in turn will help in deriving system requirements (both functional and non-functional). Functional requirements capture the intended behaviour of the system (Malan and Bredemeyer, 2001). The detailed intended behaviour of the system can be expressed as the accomplishment of intended set of behavioural (interaction) requirements with transformative functions as depicted in Figures 5.14, 5.19 & 5.20. The transformative functions are realised by internal actors that help in providing the intended effects (expected behaviour) of the system to its external actors. The exchanges and the transition in their properties from one state to another provide a more detailed perspective onto the behaviour of a system in relation to transformative functions as compared to abstract perspective onto the intended behaviour of system defined via interaction operations and exchanges at system's interfaces with its external actors. The specific relations for system architecture analysis are summarised in the UML-based class diagram in Figure 5.21. The relations are further explained in the following.

In short, following relationships are possible in between various viewpoints

- There can be many different interaction requirements within a single use case as well as there can be many common interaction requirements across different use cases;
- There can be many states' transition associated with an exchange flow;
- There can be many transformative functions in relation to exchange states' transitions;
- There can be one or more than one transformative functions associated with one use case (and vice versa);
- There can be one or more than one transformative function contributing to a single or many interaction functional and performance requirements (and vice versa);
- There can be one or more than one internal actor contributing to the transformative functions;
- There can be one or more than one internal actor that fulfil many interaction requirements associated with external actors.

5.5 Discussion

In this section, the key contributions of CM framework are highlighted in relation existing approaches for architecture analysis.

Quality function deployment is the comprehensive approach that maps the two different viewpoints; 'customer needs' into 'engineering characteristics' (i.e. 'non-functional performance specifications') in the first matrix and thereafter the 'engineering characteristics' to system's 'subsystems/internal actors' into the second matrix. Though the approach is quite robust, the definition (or distribution) of the 'functions' over the performance specifications (i.e. engineering characteristics) and components is not discussed whereas the present CM framework aims to couple 'needs (as use cases)', 'engineering characteristics (as performance requirements)' to functions (interaction and transformation) and their allocation 'subsystems/internal actors'.

Axiomatic design (Suh, 2001) provides design matrix as an approach that connects two viewpoints; the 'functional requirements' of a system to 'design parameters' and follows two axioms i.e. (i) maintain the independence of functional requirements (ii) minimise the information content. Though the axioms and design matrix provide a robust approach for evaluating design

decisions but these design parameters describe the chosen implementation (or solution) that satisfy functional requirements whereas the present CM framework aims to couple the use cases and interaction functional and performance requirements to transformation functions without implementation in mind. Furthermore, functional requirements in Axiomatic Design seem to be mainly transformative functions. The distinction between interaction functional requirements and transformative functions is not available. It is also not differentiated that which functional requirements are environment centric and device centric to a system.

Another closely related integrated requirements and functional modelling approach is proposed by Buede (2009) where 'functional requirements' on a system (including input/output and other aspects such as external interfaces) are coupled with system's 'functions'. Though, the Buede's matrix-based approach is more aligned with the present approach, it lacks to provide information from the viewpoint of 'non-functional (performance) requirements' in the same matrix which presented CM framework covers.

Bonnema (2008) provides FunKey architecting approach where stakeholders' 'key drivers' on a system are coupled with 'functions' without the implementation in mind. It also aids designers to think of budgets distribution across various system functions. Though, FunKey approach provides intuitive thinking; it does not discuss how stakeholders' drivers (i.e. performance related aspects) are derived which CM framework covers due to its integration with IAT tool where the transition is achieved from use cases to performance requirements. No distinction between types of the function is drawn in Funkey which CM framework does.

Driessen et al., (2006) shows transition of key drivers to requirements and created a matrix based approach for software architecture's description in which 'key customer drivers' for the various markets are translated (coupled) into 'functional requirements'. The functional requirements are organised and derived using a 'use cases' viewpoint. Thus it can be concluded that use cases act as a binding element between customer drivers and system functional requirements. However, their matrix-based approach, does not include coupling of 'functions' as proposed CM framework does in which 'functions' are coupled

with ‘use cases’ as well as with ‘functional and non-functional performance requirements’.

Eisenbart (2014) provides integrated function modelling approach where each matrix connects two different viewpoints but with a common viewpoint of ‘technical process’ (more like functional language). This common viewpoint, integrated with other viewpoints within functional domain closely resembles with the present framework. In integrated function modelling, the external actor’s interactions in various ‘use cases’ of the system are coupled with ‘technical processes’ in a separate matrix. Though, the integrated function modelling is a very comprehensive approach where various viewpoints within functional modelling domain are integrated robustly; it does not discuss the connectivity of requirements modelling which is covered by the proposed CM framework via an integrated IAT with it.

This section has clearly highlighted the contributions of proposed CM framework in relation to existing approaches.

5.5.1 Key conclusions

The developed CM framework is intended to provide designers and analysts with an interdisciplinary system architecting approach that is capable of integrating system’s use cases, requirements, functions, interfaces and structures’ modelling views thereby identifying coupling relationships among them. The architecture information is represented and linked in the framework by using matrix based approach and does not allow repetition of same modelling view or does not require reconfiguring a modelling view consecutively as often required by other approaches e.g. quality function deployment. In summary, the framework:

- is based on the established concept of domain mapping matrix, which links more than two different modelling views and provides compact yet structured information for a system architecture;
- is more concrete and detailed in contrast to existing modelling approaches that often bypass key view/viewpoint for architecture analysis (Section 5.5). The existing approaches lack to show structured integration with interface view which CM framework offers.

- is expected to ease and facilitate communication across disciplines, as the different views are interlinked via a central view of transformative functions;
- is flexible enough that enables addition or omission of views/viewpoints or specific information within them;
- shows clear information flow and allows traceability from system's black-box view (requirements analysis) to white-box view (physical architecture analysis).

5.6 Chapter summary

The overall aim of this chapter has been to develop a system architecting approach that links mainly interface, requirements and functions modelling views and their coupled relationships are then allocated to system's internal structure. The approach allows traceability between various modelling views that in turn involve many viewpoints. The approach shows one-to-one and one-to-many relationships among various viewpoints of various modelling views. The approach is matrix based and thus named as coupling matrix.

It is envisioned that same CM framework would work across system's various hierarchical levels. In this Chapter, IAT and CM have been discussed till one level of decomposition i.e. from system to subsystem level. In order to examine the applicability of CM framework in conjunction with IAT framework across system's multiple hierarchical levels, a real world complex industrial case study is used for validation purpose in the next Chapter.

6. Validation Case Study: Deployment of Integrated Framework to Multiple Levels of Abstraction

6.1 Introduction

This chapter initially aligns the developed integrated architecture analysis framework (Section 5.3.1.7) centred on interface modelling methodology with the key activities of systems engineering process. The integrated framework is then applied on a real world complex case within an automotive company where new technology has to be designed and integrated within the vehicle. It is shown that the integrated framework has been applied consistently across multiple levels of a new technology. The problem of the validated case study is also described followed by conclusions, discussion and results in the end.

6.2 Usefulness of IAT and CM

6.2.1 Overview of key problem in automotive industry

A key observation of the author is that system technical specialists (engineers) in automotive industry often struggle to define the right context at the right level of abstraction due to unavailability of vehicle's well-defined architecture and its hierarchical levels. Solutions (such as features or technological parts or subsystems) with good functional and logical analysis are created. However, those solutions often does not fit in the system right first time due to less focus on well-defined interfaces at the early design phase and as a result of which integration issues emerge late in the design process causing both time and money loss to an organization. McDavid (2005) emphasises the importance of robust architectural approach and stated;

> *"An architectural approach emphasizes the need for **multiple levels of abstraction**, **standardized interfaces** offered by **well-defined modules**, encapsulation, information hiding, and the like. This allows specialists to focus on **their parts of the problem**, and to meet at **well-defined interfaces**, but to understand the **other levels of the problem** that provide a **context for their work"**.*

This research proposed a practical integrated framework to engineers based on IAT and CM with other system modelling tools to define the context for the work at right level of abstraction (i.e. in the systems hierarchy) by understating the other levels work and extracting relevant information and context from them before providing a well-functioning analysis of their parts of the whole problem. Engineers need to understand and define well how their parts' interface with each other and contribute in an integrated manner for a whole problem.

In the next section, the integrated architecture analysis framework based on IAT, CM and other modelling tools is presented and discussed in the context of locating it in the systems engineering process key activities.

6.2.2 Integrated framework within the systems engineering context

The systems engineering process (Section 2.3.2.2), involves following five key activities for system design and analysis as shown in Figure 6.1.

1. Requirements analysis;
2. Functional analysis & allocation;
3. Requirements loop;
4. Design synthesis;
5. Design loop.

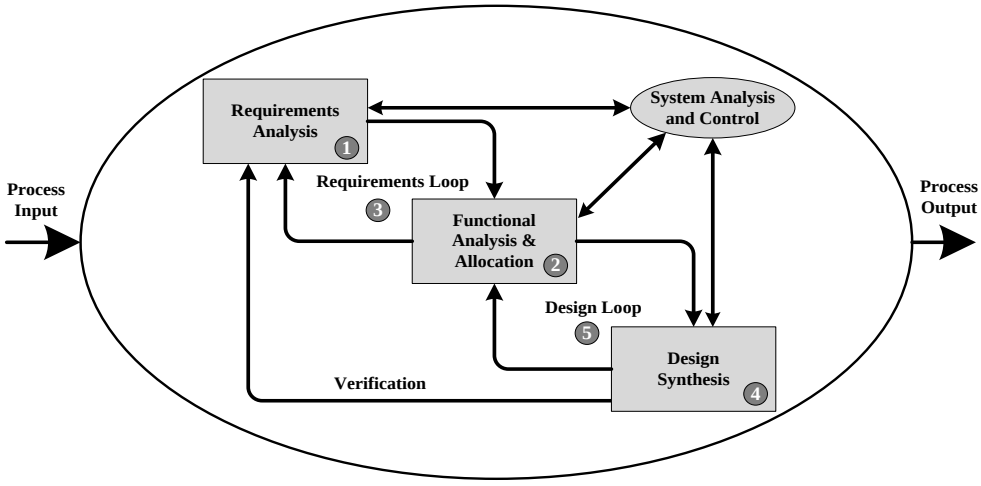


Figure 6.1 Overview of systems engineering process (adapted from DoD, 2001)

Note that this research does not focus on verification and validation activities thus those activities are not discussed as can be seen from Figure 6.1.

The developed IAT and CM are placed within the key activities of systems engineering process as shown in Figure 6.2. The 'HOW' in Figure 6.2 represents the proposed integrated architecture analysis framework comprising of series of

sets of tools that have been discussed in Chapter 5 and also belongs to FMA framework (Figure 1.1). The inputs and outputs of three block-based activities (i.e. requirements analysis, function analysis & allocation, and design synthesis) are represented along with two iterative loop activities (requirements loop and design loop) in between them that represent 'WHAT to do' for system architecture analysis. The controls (i.e. 'WHO') from top show the system architect people that would use the mechanisms located at the bottom to develop system architecture i.e. 'HOW to do' toolkit.

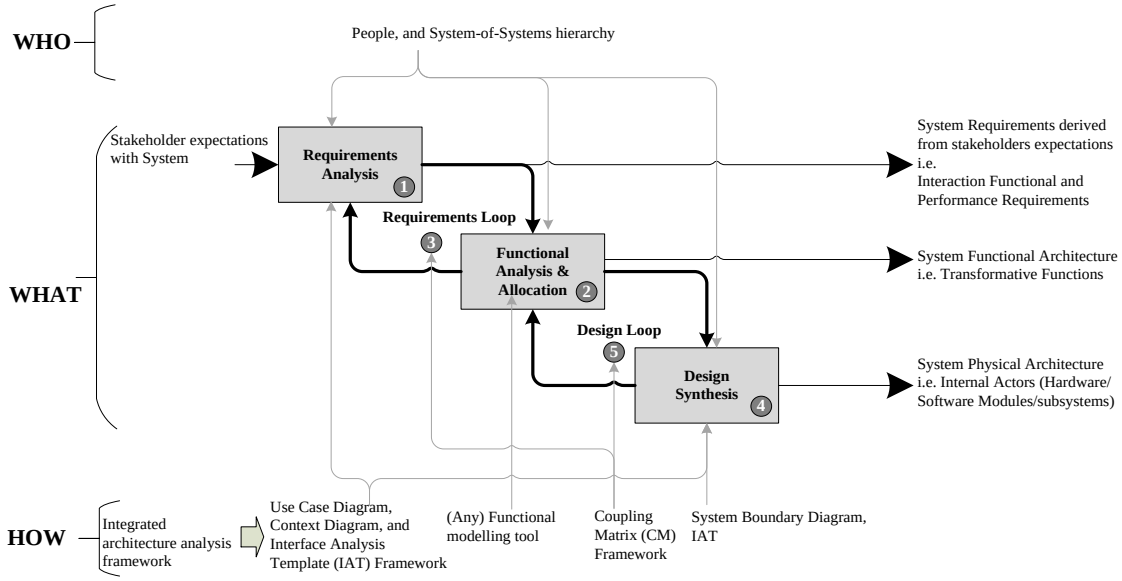


Figure 6.2 The developed architecting scheme procedural steps in systems engineering context

The integrated framework covers the 'requirements analysis' activity via a set of following complementary tools: use case diagram, context diagram, and the proposed IAT as shown in Figure 6.2. Once the functional and performance requirements set is developed via IAT in the context of a specific or multiple use cases within requirements analysis activity for a system-of-interest, then this set acts as an input to 'function analysis and allocation' activity as shown in Figure 6.2.

The 'function analysis and allocation' activity describes system functionality based on an input provided by requirements analysis thereby decomposing the higher level functions into sub-functions, resulting in a system's functional architecture as an output and subsequently allocate them to subsystems (INCOSE SEH Working Group 2004). In function analysis activity, any functional modelling tool (such as state flow diagram or function flow diagram) can be used

to develop a solution independent function architecture for a system as an output as summarized in Figure 6.2.

The ‘requirements loop’ activity, in Figure 6.2, is an iterative process in system engineering which helps in tracing the requirements between requirements and functional analysis activities. Each lower level or sub-function identified in function analysis and allocation should be traceable back to a requirement identified in requirements analysis (DoD, 2001). The requirements loop activity is accomplished via the proposed CM in which each identified transformative function is traced back to an interaction functional and performance requirement in requirements analysis, as shown in Figure 6.2. This CM, in essence, initially connects the interface requirements (performed at black-box view) with function modelling (performed at white-box view).

In Figure 6.2, similar to the requirements loop, the ‘design loop’ activity revisits the functional architecture to verify that the physical design is sufficient enough to perform the required functions and requirements at required performance level (DoD, 2001). The integrated framework also achieves this activity via later part of CM thereby allocating the mapped functions vs interfaces’ requirements to the subsystems (internal actors) that together set the platform for physical architectures development. In the final activity of ‘design synthesis’, the chosen physical architecture is further developed via system boundary diagram and detailed by IAT for synthesis purposes as illustrated in Figure 6.2.

The interface analysis in design synthesis activity (i.e. white-box view) is conducted in similar fashion as performed at system’s black-box view in requirements analysis activity which is the core strength of IAT. Similarly, the cycle of ‘HOW’ is repeated for the next decomposition level or for next critical ‘system-of-interest’ analysis at system’s subsystem level.

The integrated framework’s validation across multiple levels of a system is now presented in the next section with one complex case where new technology was introduced.

6.3 Validation case study: Regenerative braking system

The author of this research has been asked by an automotive company to support (part time) in the creation of a design and architecture for the vehicle’s braking subsystem. The collaboration required modelling of the new regenerative braking technology through diverse set of systems engineering

tools based on integrated framework underpinning IAT and CM. The automotive company also appointed a technical specialist from chassis engineering on the case study who has got years of experience and in depth knowledge both on systems engineering field, modelling tools, and braking system.

6.3.1 Case study background

Automotive companies are striving to restore maximum of the excessed energy (i.e. Wh) from kinetic movement of vehicle e.g. braking process, and their research has led to the energy recovery mechanism known as *regenerative braking technology* which is being integrated into today's vehicle. It is also often considered as one of the advanced active safety systems responsible for maintaining dynamic stability (Oleksowicz et al., 2013).

In principle, the regenerative technology slows a vehicle by converting its kinetic energy into a form which is either used immediately for some purpose or stored when required. This technology improves conventional braking system where the excess kinetic energy is converted into dissipated heat energy through friction generated in the brakes. Thus regenerative braking system also improves the efficiency of the vehicle by expanding the life of the braking system due to gradual wear of its parts.

Even though with the huge advantage of expended energy recovery, this independent system has got drawbacks in some driving circumstances (e.g. braking effect drops off at lower speeds resulting in incomplete stop of the vehicle and prevention from rolling down hills) which requires for its integration with another independent friction-based braking technology.

Note that the braking system is a subsystem within the vehicle and is quite complex enough due to its own multiple levels of abstraction and requires careful consideration of definition and management of interfaces among its subsystems and components.

6.4 The integrated architecture analysis framework's validation

The integrated framework have been applied on the regenerative braking system which is now discussed along with key insights gained in each activity. Figure 2.7 is used for referencing relevant levels of system hierarchy.

6.4.1 System-of-interest as regenerative braking system – Level 0

6.4.1.1 Identification of system context with its interaction requirements

The system context and use case diagrams were developed as a starting point to establish the context for the work of regenerative braking system. The braking system can provide a variety of use cases such as reducing vehicle speed and stopping vehicle completely. There is a potential need to extend the application of braking technological system with a new goal of 'restore expanded energy' as illustrated in Figure 6.3. In order to achieve the desired goals of braking system, its relevant interoperating actors or systems (i.e. internal stakeholders of vehicle) also need to be engaged and identified (e.g. powertrain and high voltage/power supply systems) as shown in Figure 6.3.

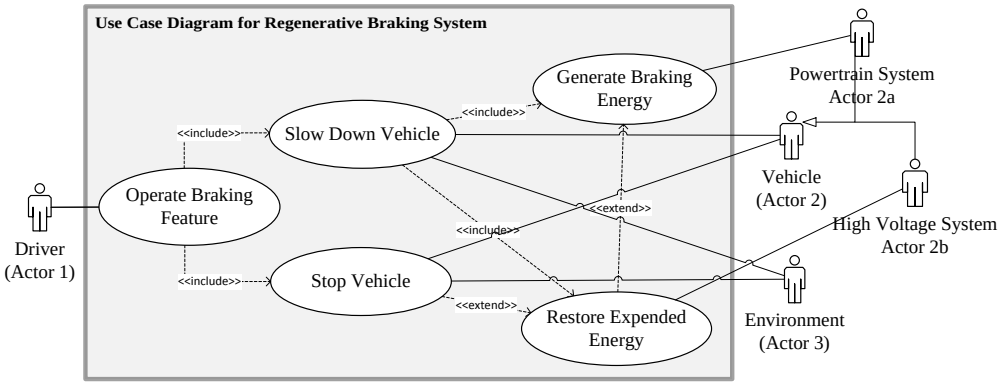


Figure 6.3 Use cases of regenerative braking system

Once the use case diagram is developed then system context diagram can also be updated/extracted as shown in Figure 6.4.

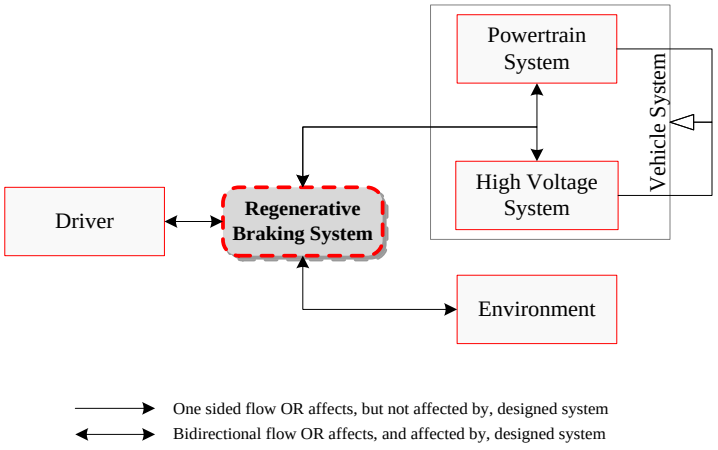


Figure 6.4 System context diagram of regenerative braking system

In the next step, the interactions between the brake system and its surrounding (external) actors are analysed. The key interaction operations and exchanges between braking system and its potential external actors (powertrain system, high voltage system, driver and environment) are identified and relevant functional and performance requirements are derived via IAT. The requirements belonging to multiple use cases are mapped in IAT. For example, functional requirements 'Deliver Force', Deliver Deceleration', and 'Receive Stroke' belong to use cases 'Slow down Vehicle' and 'Stop Vehicle' as shown in Figure 6.5. All these interaction functional requirements belong to vehicle's attribute 'A06: Brakes-Pedal Feel'. There were fifteen key interactions captured between regenerative braking systems with its four key interfaces. An excerpt of a single interface is provided in Figure 6.5 and the complete details are available in Appendix B.

Interface	Specification of Interaction Scenarios	Specification of Exchanges					Exchange Effect / Impact	System of Interest Requirements				System Use Cases	
	Interaction Operation Description	Exc. Type	Exchange Description	From	To	Exchange Properties	Attribute Type	Functional Requirement			Nonfunctional Requirement (Performance related)	UC1	UC2
								Verb	Object/Non-Object Input	Object/Non-Object Output		Stop Vehicle	Slow Vehicle
Braking System - User	101 the driver operates the braking system and gets feedback	E	Foot / Pedal force	braking system	Driver	N	A06 - brakes - pedal feel	Deliver		force	* Attribute: Force / stroke law (N / mm) Force / deceleration law (N / m/s²)	X	X
		P	Foot / Pedal stroke	Driver	braking system	mm	A06 - brakes - pedal feel	Receive	stroke		* Target value XX +/- YY	X	X
	103 The driver perceives deceleration of the vehicle	E	Vehicle deceleration	Vehicle	driver	m/s2	A06 - brakes - pedal feel	Deliver		deceleration	* Repeatability	X	X
										deceleration			

Figure 6.5 An excerpt of requirements of a braking system via IAT at black-box

Observations and lessons learnt: On one hand, from company's perspective, the automotive engineers found the incredible effectiveness of IAT as it helped them in identifying new requirements and redefining existing requirements from input/output perspectives step by step in detail manner. They also observed that IAT provides more concrete information and reflects clear operational and behavioural aspects.

On the other hand, it is observed by the author during assistance on the case study that a single performance requirement can be a combination of a pair of interaction functional requirements. For example, 'Deliver Force' and 'Receive Stroke' share same performance requirements i.e. 'Force/Stroke' and 'Force/deceleration' which are key measurable attributes in the regenerative braking system analysis as can be seen in Figure 6.5. The information was managed easily due to flexible structure of IAT.

6.4.1.2 Identification of transformative functions of the regenerative braking system

The transformative functions have been identified through a number of following functional modelling tools.

- Activity diagram (a behavioural modelling tool in SysML)
- Functional flow diagram
- System state flow diagram

A use case from use case diagram can be articulated in engineering language. For example, a top-level function equivalent to a use case ‘Slow down vehicle’ (Figure 6.3) would be ‘Decelerate Vehicle’ in engineering language (see Appendix B). The sub-functions belonging to this use case or top-function have been identified using functional flow diagram as shown in Figure 6.6. The system state flow diagram was also produced and is available in the Appendix B.

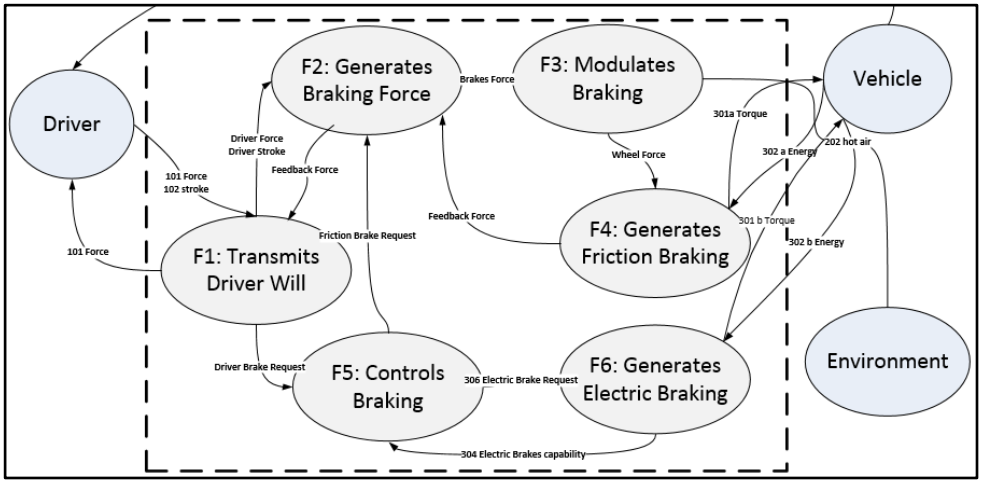


Figure 6.6 An excerpt of functional architecture of regenerative braking system

Observations and lessons learnt: It was observed that most of the functional modelling tools provide more or less common functions with a key difference of heuristics around a functional modelling tool. It was also observed that system state flow diagram tool (Figure 2.17) was found to be hardest one due to its careful thinking and definition around states’ properties and transitions. Granted the inventory of functions remain mostly similar they found that it takes more time than the other functional modelling tools.

6.4.1.3 Allocation of interaction requirements to transformative functions of the regenerative braking system

In the next step, transformative functions that contribute to fulfil the system's interaction functional and performance requirements are then mapped out via CM as shown in Figure 6.7. For example, both transformative functions 'Transmit Driver demand' and 'Generate Braking force' are required for respective use cases 'Stop Vehicle' and 'Slow down Vehicle'. Also both transformative functions are responsible for delivering the interaction functional (i.e. 'Receive Stroke' and 'Deliver mechanical Force') and relevant performance requirements (e.g. Force per Stroke law with target value X < N/mm < Y). An excerpt of CM is provided in Figure 6.7 and the complete CM is available in Appendix B.

									Use Cases	Restore Expended Energy			
										Slow Vehicle (Electric Braking)	X	X	
										Stop Vehicle (Fricton Braking)	X		
Interface	**Description of Interaction Operation**	**Description of Interaction Exchange**				**Exchange Effect Impact**	**System of Interest Requirement and Specification**					**Transformation Functions**	
	Description of Interaction	**Type (P, M, E, I)**	**Exchanges: Object/ Noun Ouput**	**From**	**To**	**Attribute type**	**Interaction Functional Requirement**			**Requirement Specifications (Attribute + Target Value + Relation+ Unit)**		**F1**	**F2**
							Verb	**Object/Non-Object**				Transmit Driver Will	Generate Braking Force
								Input	**Output**				
Braking System - User	101 the driver operates the braking system and gets feedback	E	Foot / Pedal force	braking system	Driver	A06 - brakes - pedal feel	Deliver		force	* Attribute: Force / stroke law (N / mm) Force / deceleration law (N / m/s²) * Target value XX +/- YY * Repeatability		X	X
		P	Foot / Pedal stroke	Driver	braking system	A06 - brakes - pedal feel	Receive	stroke				X	X
	103 The driver perceives deceleration of the vehicle	E	Vehicle deceleration	Vehicle	driver	A06 - brakes - pedal feel	Deliver		deceleration			X	X
	201 the braking system dissipates heat in the	E	Heat dissipation	braking system	Enviro nment	A06 - brakes - performance	Minimis e		heat dissipation	KJ (target = energy to be dissipated to reduce 302a			

Figure 6.7 Requirements loop analysis for regenerative braking system via CM

6.4.1.4 Identification of internal actors/design solutions

Once, it is understood which transformative functions of regenerative braking system contribute to which interaction requirements (of which external actors) in a use case or use cases of a system, then in the next step, its physical architecture is created. It is the stage of defining hardware and software elements that together make a regenerative brake system such as actuation, foundation and high level control subsystems at white-box as shown in Figure 6.8.

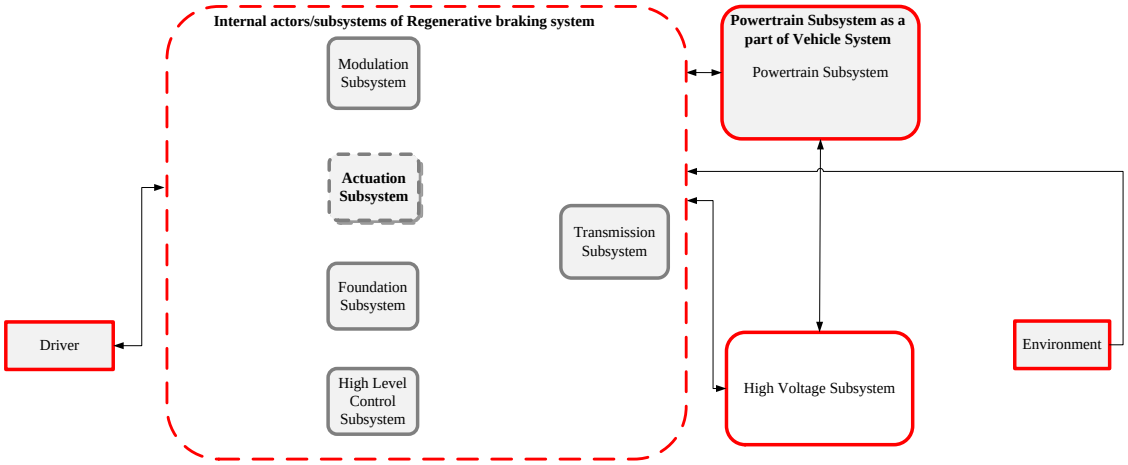
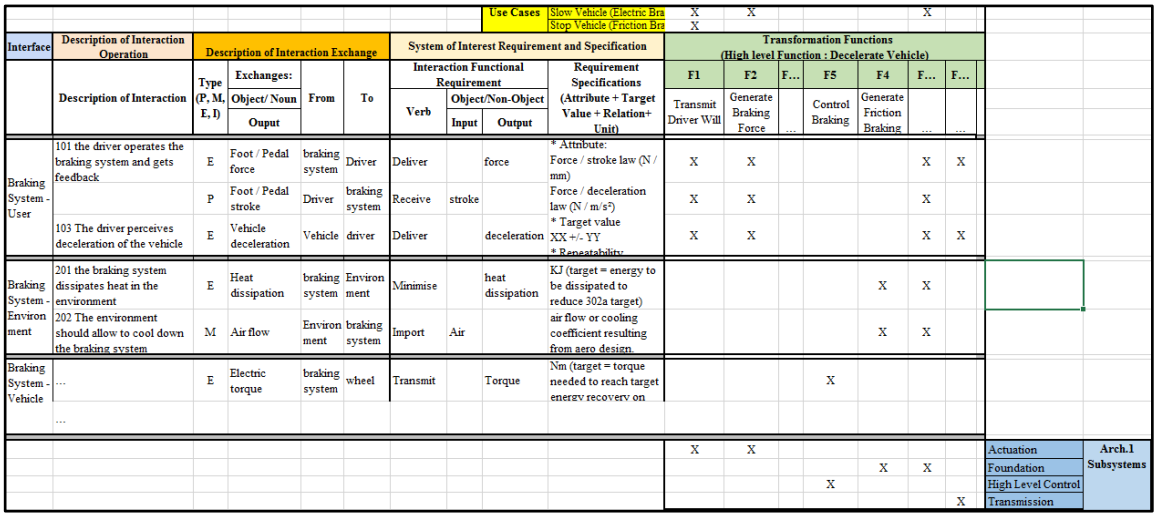


Figure 6.8 Internal actors/subsystems of regenerative braking system

6.4.1.5 Allocation of transformative functions vs interaction requirements to subsystems/internal actors

In the next step, the CM was expanded to allocate the transformative functions vs interaction requirements to regenerative braking system's internal actors/subsystems. Note that a single function can be achieved by more than one subsystem and vice versa.

For example, in Figure 6.9, a single subsystem 'actuation system' serves functions 'F1: Transmit Driver Demand' and 'F2: Generate Braking Force'. The CM helped the engineers to understand and allocate the single or multiple functions to subsystems but also provided information that what interaction functional and performance requirements are associated with these transformative functions that the subsystems should meet to deliver the whole system's targets and objectives. For example, along with the delivery of transformative function 'F1:Transmit Driver Demand' by actuation system, it must meet a set of interaction functional requirements such as 'Deliver Force', and 'Receive Stroke' within the acceptable limits of measurable performance requirements i.e. 'Force/stroke' which are tightly coupled as can be seen in Figure 6.9.



								Use Cases	Slow Vehicle (Electric Bra	X	X				X					
									Stop Vehicle (Friction Bra	X										
Interface	Description of Interaction Operation	Description of Interaction Exchange				System of Interest Requirement and Specification				Transformation Functions (High level Function : Decelerate Vehicle)												
	Description of Interaction	Type (P, M, E, I)	Exchanges: Object/ Noun Ouput	From	To	Interaction Functional Requirement Verb	Object/Non-Object Input	Object/Non-Object Output	Requirement Specifications (Attribute + Target Value + Relation+ Unit)	F1	F2	F…	F5	F4	F…	F…					
										Transmit Driver Will	Generate Braking Force	…	Control Braking	Generate Friction Braking	…	…					
Braking System - User	101 the driver operates the braking system and gets feedback	E	Foot / Pedal force	braking system	Driver	Deliver		force	* Attribute: Force / stroke law (N / mm)	X	X				X	X					
		P	Foot / Pedal stroke	Driver	braking system	Receive	stroke		Force / deceleration law (N / m/s²)	X	X				X						
	103 The driver perceives deceleration of the vehicle	E	Vehicle deceleration	Vehicle	driver	Deliver		deceleration	* Target value XX +/- YY * Repeatability:	X	X				X	X					
Braking System - Environment	201 the braking system dissipates heat in the environment	E	Heat dissipation	braking system	Environment	Minimise		heat dissipation	KJ (target = energy to be dissipated to reduce 302a target)					X	X						
	202 The environment should allow to cool down the braking system	M	Air flow	Environment	braking system	Import	Air		air flow or cooling coefficient resulting from aero design.					X	X						
Braking System - Vehicle	…	E	Electric torque	braking system	wheel	Transmit		Torque	Nm (target = torque needed to reach target energy recovery on				X								
	…																					
										X	X								Actuation	Arch.1 Subsystems		
														X	X				Foundation			
													X						High Level Control			
																X			Transmission			

Figure 6.9 An excerpt of architecture of regenerative braking system till level-1 via CM

Observation: It was observed that engineers named generic subsystems of regenerative braking system due to their experience on existing technologies but still they did not choose any specific subsystem/module (or internal actor) solution in mind as they did not design those yet. At this stage, they wanted to understand how these modules in general can behave or would interact to each other to meet braking system requirements. The scope for further analysis zoomed into actuation subsystem/module which was considered pivotal for regenerative braking technology and for energy recovery objective in contrast to other subsystems. It was equally important to understand at first place how this actuation subsystem must fit and interact with the other subsystems of a braking system such as foundation subsystem (a friction-based technology) before zooming into its internal functionality.

6.4.1.6 Analyse interfaces of a chosen architecture

At this stage the abstract relationships between the subsystems via system boundary diagram are defined as shown in Figure 6.10 and further detailed and analysed via IAT for understanding the interaction information in between them, as shown in Figure 6.11. Now the interface requirements among the subsystems need to be captured but from both subsystems' side-views (i.e. pairwise) (as discussed in Section 4.3.2).

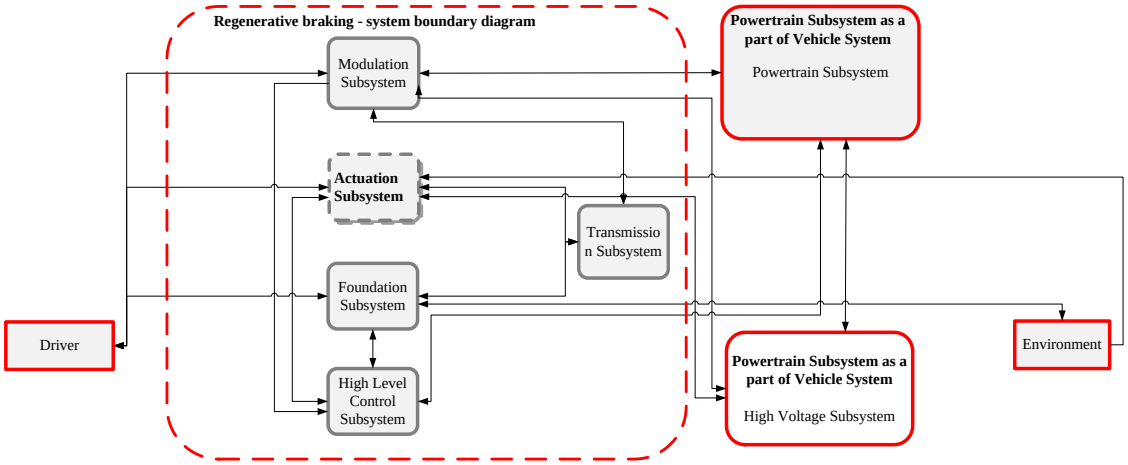


Figure 6.10 Regenerative braking system boundary diagram (white-box)

At this stage, following two cases were observed:

- ***Requirements to capture between black-box and white-box actors (i.e. internal to external actors' interactions)***

  New or additional interaction requirements may appear between external and internal actors' interfaces OR may require carry over of interaction requirements from black-box to white box. For example, some carry over interaction functional requirements 'Receive Stroke' and 'Deliver Force' allocated to actuation system from black-box remained same as an interaction requirements between actuation system and driver actor at white-box, as shown in Figure 6.11.

Interface Requirements Systems of Braking System	Interface		Description of Interaction	Description of Interaction Exchange				Exchange Effect Impact	Interaction Requirements and Specification between subsystems of Braking System			F: Decelerate Vehicle	
									Functional Requirement			F2	F5
			Interaction Operation	Exchange Type (P, M,	Exchanges: Object/ Noun	From	To	Atribute type	Verb	Object / Non-Object Input	Output	Generate Braking Force	Control Braking
	Internal to External Actors/Subsystems' Interactions												
	Driver - Actuation Subsystem		The driver appplies foot force and gets feedback	E		Driver	actuation	A06 pedal feel / brakes	Deliver		Force	X	
				E/P		Driver	actuation	A06 pedal feel / brakes	Receive	Stroke		X	
	Brakes High Level Control Subsystem - Powertrain System		Powertrain requests for braking status from High Level Control Subsystem	I	Braking Force Information	Brakes high level control	Powertrain	Energy management	Import	Braking Status Information			
				I		Brakes high level control	Powertrain		Transmit		Braking Status Information		X
			...										
	Internal to Internal Actors/Subsystems' Interactions												
			The actuation system requests for		boost	brakes high		A06 pedal feel /		Boost			

Figure 6.11 An excerpt of subsystems' interface requirements of brake system

- ***Requirements to capture between the white-box' actors (i.e. internal to internal actors' interactions)***

  Similarly, the interaction requirements of regenerative braking system's subsystems interfaces at white-box are identified and detailed in IAT, as

shown in Figure 6.12. For example, the ‘actuation subsystem’ and ‘high level control subsystem’ share ‘boost command’ and ‘actuation status’ exchanges related requirements. The pair wise interaction requirements from each system’s side view is captured in IAT. For example, the input interaction functional requirement from actuation subsystem’s perspective is ‘Receive boost command’ and same exchange based requirement is the output from control subsystem’s perspective as ‘Export boost command’ and the directionality is from ‘high level control subsystem’ to ‘actuation subsystem’.

The interfaces filled with yellow colour cells in Figure 6.12 indicate interaction requirements articulated from actuation system perspective while green colour ones from other subsystems perspectives.

It should also be noted that each subsystem interaction requirements are also linked with the regenerative braking system transformative functions that were allocated to them. For example, in Figure 6.12, in an interface of ‘actuation - high level control’ subsystems, the actuation subsystem’s interaction functional requirement ‘Receive boost command’ is linked with braking system’s transformative function ‘Generate Braking Force’ (which was allocated to actuation system in CM - see Figure 6.9). Similarly, the control subsystem’s interaction functional requirement ‘Export boost command’ is linked with braking system’s transformative function ‘Control Electrified Torque’.

Interface Requirements between Subsystems of Braking System

Interface		Description of Interaction	Description of Interaction Exchange				Exchange Effect Impact	Interaction Requirements and Specification between subsystems of Braking System			F: Decelerate Vehicle	
								Functional Requirement			F2	F5
		Interaction Operation	Exchange Type (P, M,	Exchanges: Object/ Noun Ouput	From	To	Atribute type	Verb	Object / Non-Object Input	Object / Non-Object Output	Generate Braking Force	Control Braking
Internal to External Actors/Subsystems' Interactions												
Driver - Actuation Subsystem		The driver appplies foot force and gets feedback	E		Driver	actuation	A06 pedal feel / brakes performance	Deliver		Force	X	
			E/P		Driver	actuation	A06 pedal feel / brakes performance	Receive	Stroke		X	
Brakes High Level Control Subsystem - Powertrain System		Powertrain requests for braking status from High Level Control Subsystem	I	Braking Force Information	Brakes high level control	Powertrain	Energy management	Import	Braking Status Information			
			I		Brakes high level control	Powertrain		Transmit		Braking Status Information		X
		…										
Internal to Internal Actors/Subsystems' Interactions												
Brakes high level control - Actuation Subsystem		The actuation system requests for boosting when required	I	boost command	brakes high level control	actuation	A06 pedal feel / brakes performance	Receive	Boost Command		X	
								Export		Boost Command		X
		The actuation sub-system provides its status to the brakes high level control sub-system	I	actuation status	actuation	brakes high level control	A06 pedal feel / brakes performance	Transmit		Actuation Status	X	
								Receive	Actuation Status			X
Actuation Subsystem - Vacuum source		The actuation sub-system receives vacuum from the vacuum source	P	vacuum	vacuum source	actuation	A06 pedal feel	Import	Vacuum		X	
								Export		Vacuum		

Figure 6.12 IAT between internal and external actors of regenerative braking system

6.4.1.7 Repeat cycle for next critical system-of-interests

Once, the system level analysis are completed both at level 0 - black-box (see Figures 6.4 and 6.5) and level 1 - white-box views (see Figures 6.10 and 6.12), then the next critical system-of-interest's architecture analysis can be conducted by using the system level information available at white-box. For example, actuation system was considered as a next critical system-of-interest for further analysis in braking system.

The actuation system which is a white-box for its braking system in turn becomes a black-box for other subsystems of a braking system at the same level. This means that the interaction requirements captured at braking system's white-box become black-box requirements for actuation system. This fact is completely aligned with Crilly's (2012) nested systems concept in which the device centric functions of a system (e.g. see regenerative braking system's interaction functional requirements in Figure 6.12) become environment centric functions for its subsystem (e.g. actuation system).

Therefore, actuation system's context and its black-box interface requirements are drawn from braking system's white-box analysis. Thus, it is important to note and recognise here that system-of-interest's white-box interaction analysis become black-box analysis for its sub-system-of-interest.

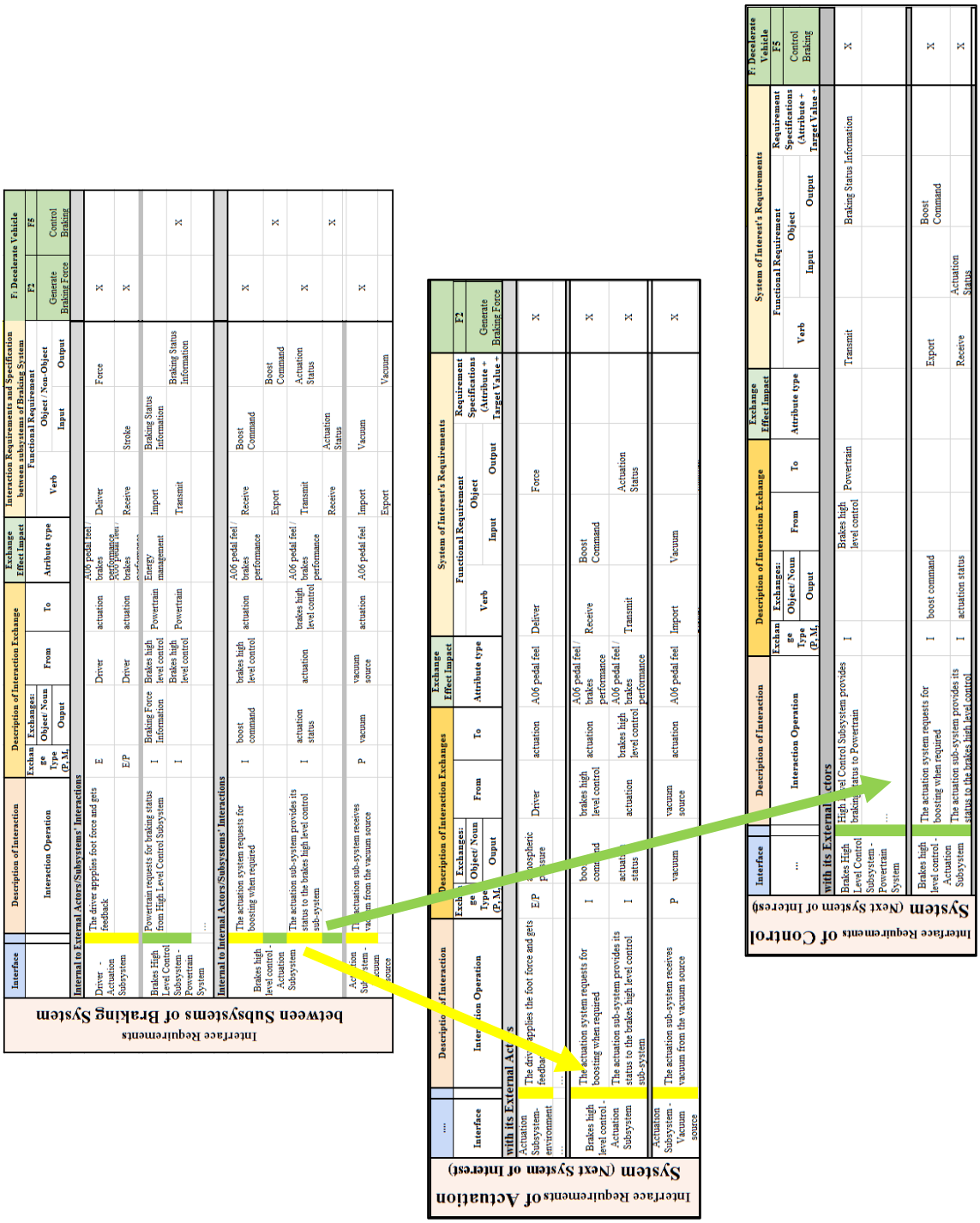


Figure 6.13 An excerpt of subsystems' interaction requirements drawn from regenerative brake system's white-box IAT (i.e. Figure 6.12)

6.4.2 System-of-interest as actuation system – Level 1

6.4.2.1 Identification of interaction requirements on actuation system

Given that system boundary diagram, in Figure 6.10, and IAT in Figure 6.12 of a brake system; the available information in these templates is then used to define the context of the actuation system thereby extracting its requirements

from regenerative braking system's IAT as illustrated in Figure 6.13. Similarly, the context diagram of actuation system can also be derived with its external actors from regenerative braking's system's boundary diagram, in Figure 6.10, as shown in Figure 6.14. Note that at this stage it is not known what type of architecture of actuation system can meet the requirements drawn for it as shown in Figure 6.15 as there are many architectures of actuation system. This demands for the analysis of its functional architecture which is the next step in the scheme.

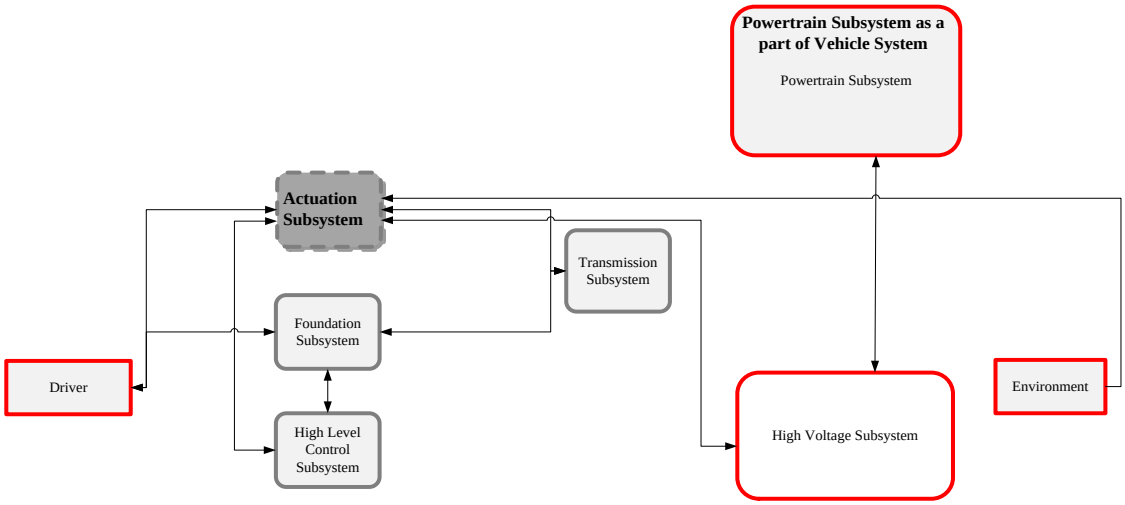


Figure 6.14 Actuation system's context diagram extracted from regenerative system's boundary diagram

Interface Requirements of Actuation System (Next System of Interest)	Interface	Description of Interaction	Description of Interaction Exchanges				Exchange Effect Impact	System of Interest's Requirements				
	....	Interaction Operation	Exchange Type (P, M, E, I)	Exchanges: Object/ Noun Ouput	From	To	Attribute type	Functional Requirement: Verb	Object: Input	Object: Output	Requirement Specifications (Attribute + Target Value +	F2 Generate Braking Force
	with its External Actors											
	Actuation System - Driver	the driver applies a foot force and gets a feedback	E	force	driver	actuation	A06 pedal feel	Deliver		Force		X
			P	stroke	driver	actuation	A06 pedal feel	Receive	Stroke			X
	Actuation Subsystem- Power Supply	The actuation system requires electric energy to operate	E	operating energy	High Voltage/ power supply	actuation	A06 pedal feel / brakes performance	Accept	Electric Power			X
	Brakes high level control - Actuation Subsystem	The actuation system requests for boosting when required	I	boost command	brakes high level control	actuation	A06 pedal feel / brakes performance	Receive	Boost Command			X
		The actuation sub-system provides its status to the brakes high level control sub-system	I	actuation status	actuation	brakes high level control	A06 pedal feel / brakes performance	Transmit		Actuation Status		X
	Actuation Subsystem - Vacuum source	The actuation sub-system receives vacuum from the vacuum source	P	vacuum	vacuum source	actuation	A06 pedal feel	Import	Vacuum			X

Figure 6.15 Requirements of actuation system at black-box from regenerative brake system white-box analysis

6.4.2.2 Identification of transformative functions of the actuation system

The functional structure of actuation system was then developed by decomposing the top level function ‘Generate Braking Force’ (Figure 6.6) as shown in Figure 6.16. For brevity reasons, only an excerpt related to actuation system is provided.

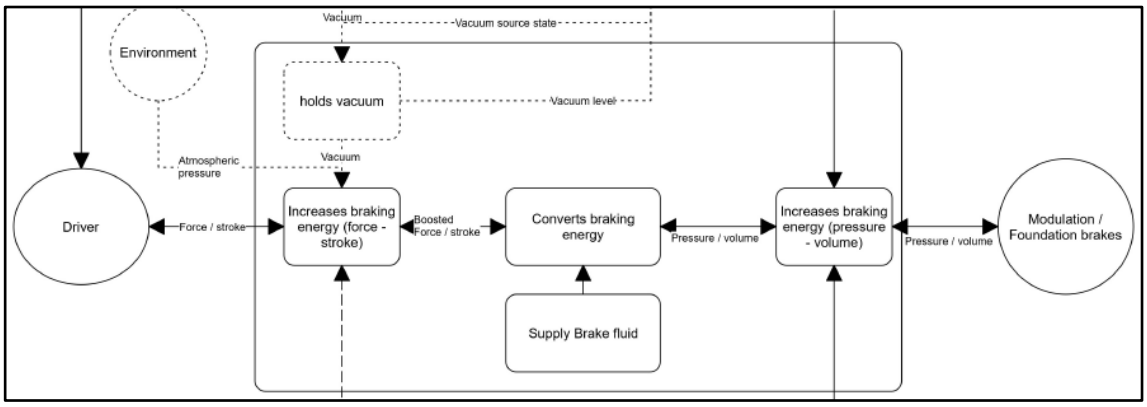


Figure 6.16 Functional architecture of actuation system

6.4.2.3 Allocation of interaction requirements to transformative functions of the actuation system

The requirements loop activity is then performed between actuation system’s interaction requirements and transformative functions. The sub-functions inventory of actuation system in Figure 6.16 of high level function ‘Generate braking force’, is then coupled to requirements inventory of Figure 6.15, via CM framework as shown in Figure 6.17.

Actuation System Interface Requirements	Interface	Description of Interaction	Description of Interaction Exchange				Exchange Effect Impact	System of Interest Requirement and Specification			F2: Generate Braking Force				
				Exchanges:				Functional Requirement			F24	F21	F22	F25	F23
	...	Description of Interaction Operation	Exchange Type (P, M, E, I)	Object/ Noun Ouput	From	To	Attribute type	Verb	Object/Non-Object Input	Output	Holds Vacuum	Increase Braking Energy (Force-Stroke)	Converts Braking Energy	Supply Brake Fluid	Increase Braking Energy (Pressure-Volume)
	Actuation System Interactions with Brake System's External actors														
	Actuation System- Driver	the driver applies a foot force and gets a feedback	E	force	driver	actuation	A06 pedal feel	Deliver		Force					
			P	stroke	driver	actuation	A06 pedal feel	Receive	Stroke			X			
	Actuation System - Power Supply	The actuation system requires electric energy to operate	E	operating energy	power supply	actuation	A06 pedal feel / brakes performance	Accept	Electric Power			X			X
	Actuation System interactions with other Brakes System's Internal Actors											X	X		X
	Actuation system - Brakes high level control	The actuation system requests for boosting when required	I	boost command	brakes high level control	actuation	A06 pedal feel / brakes performance	Receive	Boost Command						X
		The actuation sub-system provides its status to the brakes high level control sub-system	I	actuation status	actuation	brakes high level control	A06 pedal feel / brakes performance	Transmit		Actuation Status		X			X
	Actuation System - vacuum source	The actuation sub-system receives vacuum from the vacuum source	P	vacuum	vacuum source	actuation	A06 pedal feel	Import	Vacuum		X	X			

Figure 6.17 Requirements loop analysis for actuation system via CM

6.4.2.4 Identification of internal actors/design solutions

The multiple architectures were identified that could deliver a solution independent function structure developed for actuation system in Figure 6.16 and could also meet its interaction requirements in Figure 6.15. This point became quite evident at this stage. Figures 6.18 and 6.19 show three possible physical architecture for an actuation system that can be implemented to achieve the expected coupled interaction requirements and functions (Figure 6.17).

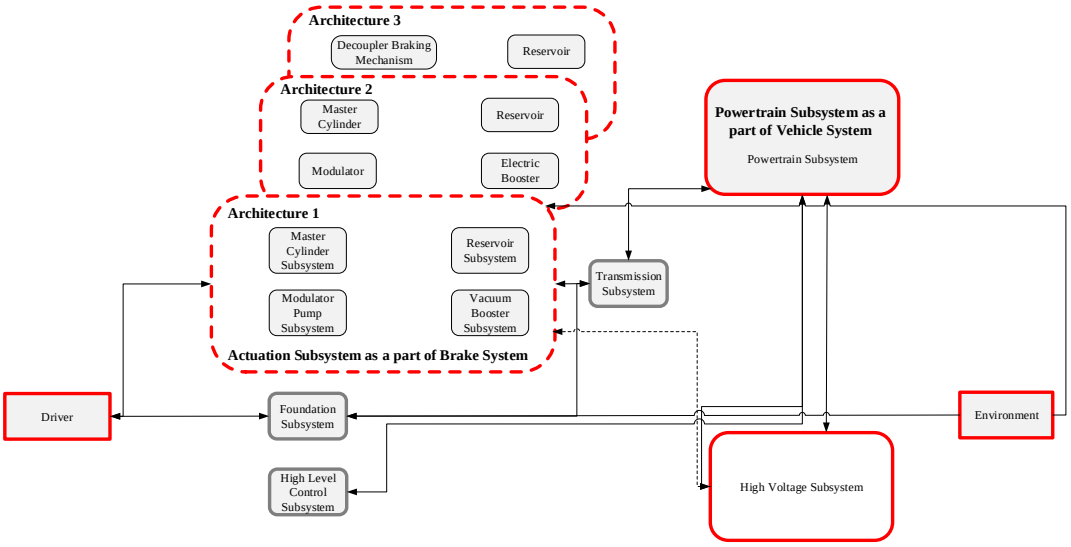


Figure 6.18 System boundary diagram of actuation system

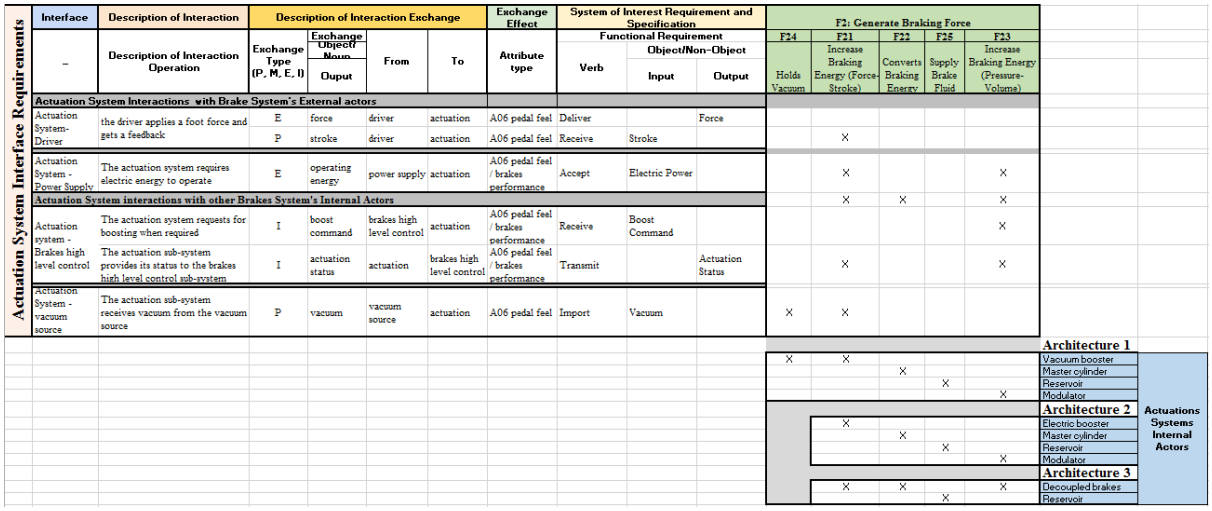


Figure 6.19 System architecture with design loop of actuation system

The CM created at this stage provided valuable insights as follows:

- Note in CM, in Figure 6.19, that one architecture (i.e. Arch. 1) fulfils all sub-functions of actuation system whereas the other two architectures (i.e. Arch. 2 & 3) fulfil four sub-functions. The key point to note is that CM indicates that all these three architectures can meet actuation system's requirements.
- The single sub-function 'Generate Braking Energy' is served by three different technical concepts in each architecture: (1) Vacuum Booster in Arch. 1, (2) Electric Booster in Arch. 2, and (3) Decoupled Braking Mechanism in Arch. 3.
- The 'Vacuum booster' component in Arch. 1 require two transformative sub-functions 'Hold Vacuum' and 'Generate Braking Energy' to meet an interaction functional requirement 'Import Vacuum' of actuation system as shown in Figure 6.19. On the other hand, 'Electric Booster' component in Arch. 2 and 'Decouple Brakes' component in Arch. 3 require only 'Generate Braking Energy' sub-function to meet same interaction functional requirement 'Import Vacuum'. The conclusion is that different architectures can be generated that may partially or fully satisfy solution independent transformative functions but important is that those meet and deliver same interaction functional requirements.
- The other key point to note, in Figure 6.19, is that Arch. 2 and Arch. 3 are of two different architecture types i.e. one case is modular architecture type (i.e. Arch. 2) and the other case is of integrated architecture type (i.e. Arch. 3). In modular architecture, system's subsystems are functionally self-contained in one-to-one relationships i.e. one function is delivered by one subsystem which is the case in Arch. 2. In integrated architecture, subsystems are functionally tightly coupled in one-to-many relationships i.e. multiple functions are delivered by a single subsystem and vice versa which is the case in Arch. 3. Recall that one of the requirements on a novel system architecting approach was that it should support and manage both modular and integrated architectures creation (see Section 3.6.2.1 bullet point no. 4 in Chapter 3) which the developed CM does. The desktop examples considered in Chapter 5 were more of integrated architecture type whereas automotive engineers aimed to create modular architectures.

6.4.2.5 Analyse interfaces of a chosen architecture

The physical architecture out of three architectures of actuation system is then chosen based on trade-off criteria by engineers. The cycle is repeated for next critical subsystems/components of actuation system. The complete system boundary diagram for braking system hierarchy including actuation system in it is shown in Figure 6.20. It can be stated that Figure 6.20 represents system-of-systems thinking (Figure 2.1) and hierarchical structure of regenerative braking system within a vehicle.

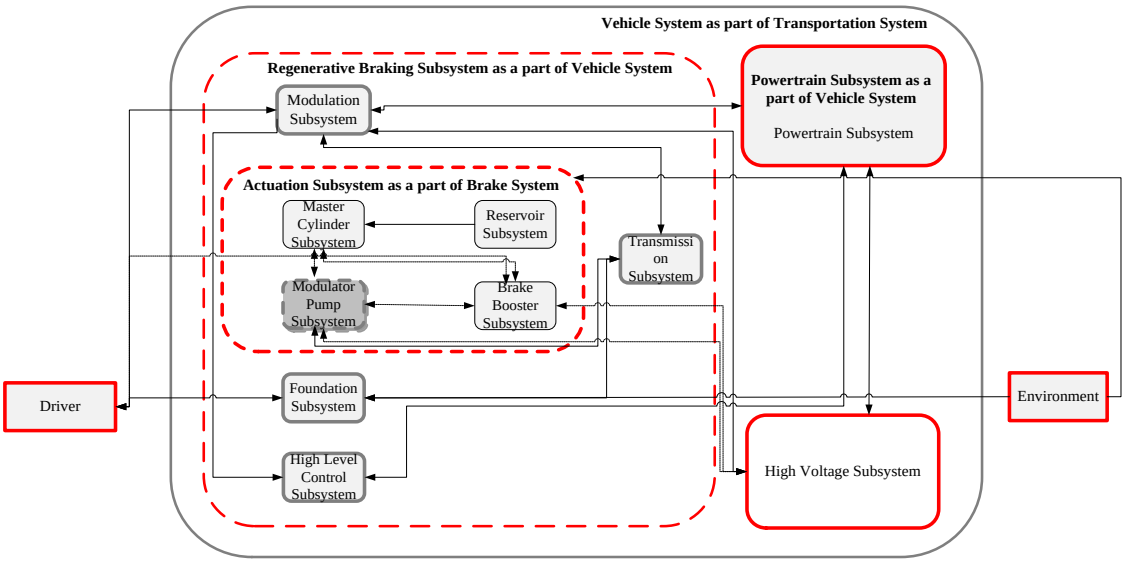


Figure 6.20 System-of-systems representation for a braking system hierarchy via boundary diagram

6.4.3 Discussion

It is shown in this chapter that system-of-systems thinking applied via the integrated framework on a complex braking technology. The CM's excerpt, in Figure 6.21, shows the overview of traceability and maintenance of requirements top-down from system (i.e. braking system) to subsystem (actuation, control system etc.) to components (master cylinder, pump etc.) across two levels of abstraction and also bottom-up from components to subsystem to system.

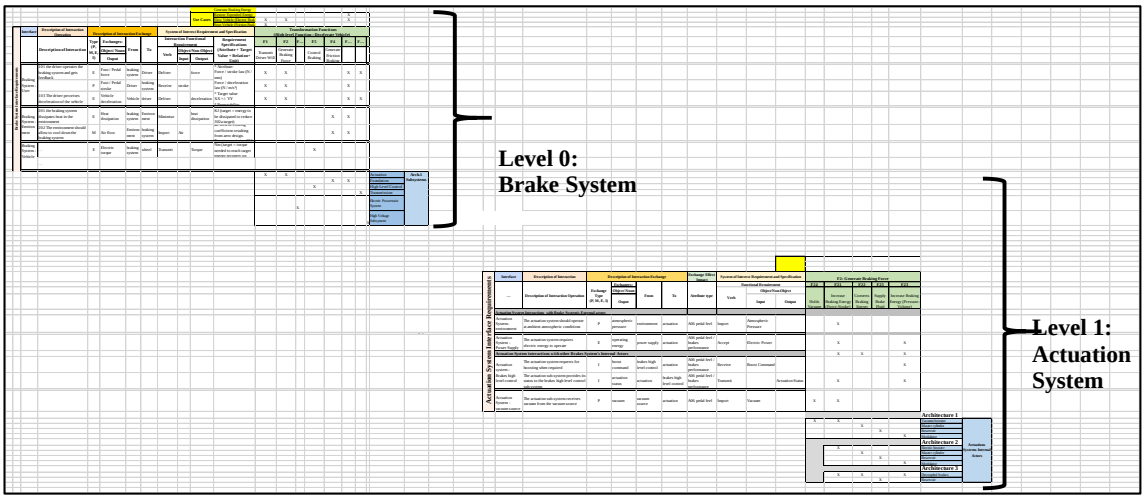


Figure 6.21 Integrated framework information flow from regenerative braking system to actuation sub-system to vacuum booster component

The function tree structure extracted from level 0 – level 1 (Figure 6.21) is organised in the context of (1) device-centric and environment-centric concepts & (2) solution dependent and independent analysis concepts as shown in Figure 6.22.

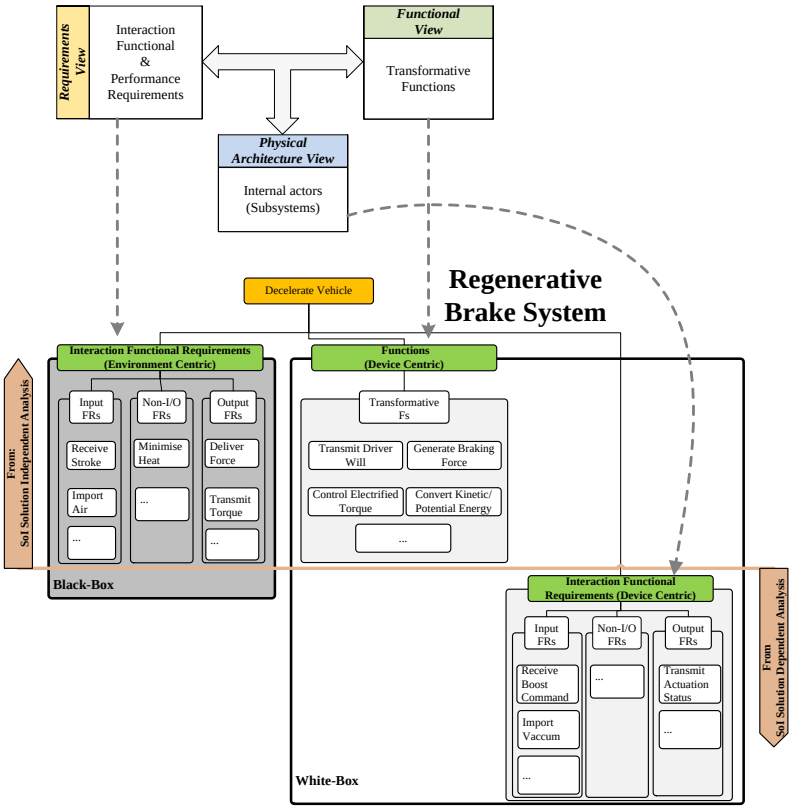


Figure 6.22 Function tree of the regenerative braking system at one level decomposition

The function tree, in Figure 6.22 shows the clustering of transformative functions and interaction functional requirements into environment centric and device centric concepts for regenerative braking system. The solution dependent and independent analysis at one level of abstraction is also shown in Figure 6.22. The relevant clustered groups in Figure 6.22 are also mapped with relevant blocks of CM (Figure 5.6). Figure 6.23 reflects the fact that the device centric's interaction functional requirements within a system at one abstraction level (e.g. regenerative braking system) become environment centric interaction functional requirements for its next critical subsystem at next lower level (e.g. actuation subsystem).

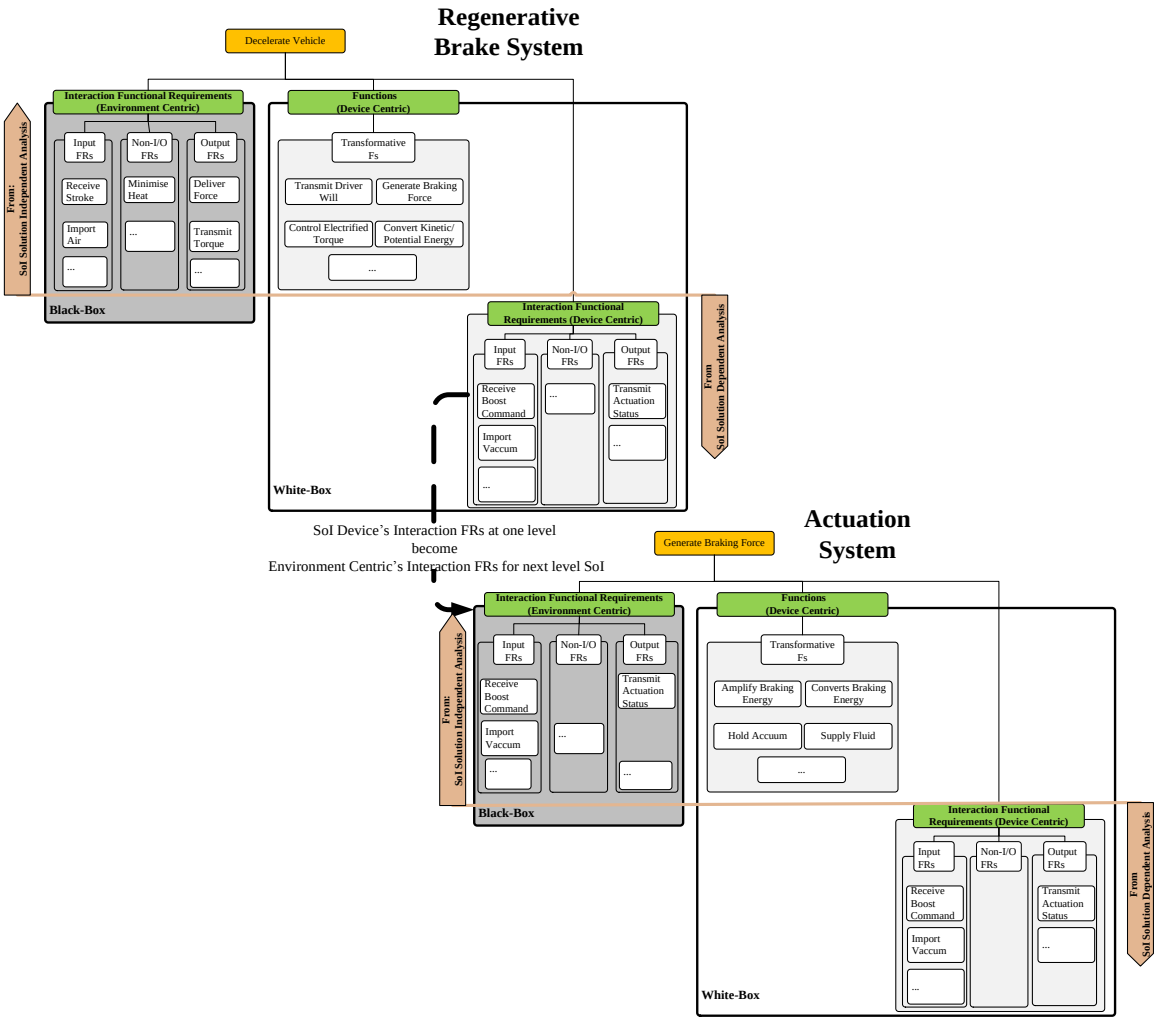


Figure 6.23 Regenerative braking system's functions hierarchy across multiple levels decomposition

Figure 6.24 shows that use cases of a system-of-interest are usually specified at the top of the pyramid at an abstract level which is aligned with high level abstraction concept of Tomiyama's top-down pyramid. The requirements are then derived by analysing the use cases (goals) and the interaction operations of stakeholders with system which are transformed into measurable requirements via IAT at system level. These requirements are then cascaded or pass down via CM to subsystems, and components where further detailed design and interface specifications in IAT for subsystems/components in subsequent levels are defined. The number of design elements (subsystems/components) increase with levels of abstraction (i.e. 10^0 to 10^6) as indicated on vertical axis of the pyramid in Figure 6.24. At the bottom of pyramid, complexity is even higher but mostly mono-disciplinary (i.e. hardware or software, etc.). For example, control subsystem in brake system can be conceptualised analogous to software discipline and actuation system to hardware discipline. Similarly, the number of requirements increases from top of pyramid to bottom of pyramid due to number of issues or problems emerging at each successive level along with increase in number of interfaces and interactions to manage at lower levels. These are tightly coupled and thus IAT in conjunction with CM framework help in managing the system's abstraction complexity from top to bottom by strictly following system-of-systems thinking.

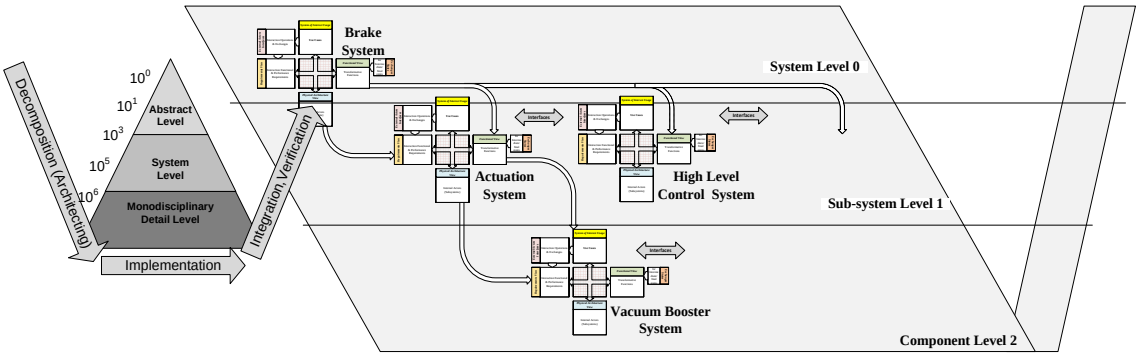


Figure 6.24 The architecture and requirements analysis in the context of top-down flow on left side of V-model (aligned with Tomiyama (2012) pyramid)

6.4.4 Key conclusions

The validation case study has revealed several benefits to industrial technical specialists during the implementation of the integrated architecture analysis

framework. The integrated framework was new to the technical specialists however its implementation led to following key conclusions:

- Abstraction levels are difficult to establish which are quite important for defining the right context for a system-of-interest and locating it in the hierarchy. The integrated framework based on IAT and CM helps in creating, and managing multiple levels of abstraction.
- Definition of well-defined interfaces for a system-of-interest is absolutely pivotal for system architecting activities such as decomposition and integration activities. The IAT not only helps in robust definition of the system-of-interest interfaces but also supports system architecture analysis.
- Creation of system-of-interest's multiple architectures (both modular and integrated types) during the introduction of new technology is beneficial given trade-off criteria is well established early in the design stage. The CM supports in establishing and managing multiple architectures information.
- Software prototype need was felt for the integrated framework as currently it is document based. It took slightly more time by automotive engineers than expected by the researcher. The tools in integrated framework including CM and IAT are of various formats such as Microsoft excel sheets, and visio format based.

During the validation process with industrial partner, the integrated framework was applied on three different real world multidisciplinary case studies. The concepts were appreciated with positive feedback which will be discussed in next section in detail. The integrated framework based on IAT and CM showed promising results. Overall, the effectiveness of the integrated framework was appreciated in following three key aspects; (1) helped in identifying missing requirements at interfaces (i.e. from completeness perspective) that escaped through current process due to more focus on structural/production related interactions, (2) multiple viewpoints linkage and requirements allocation clarity and (i.e. concreteness) (3) finally, the need of defining right hierarchical levels and requirements cascading across multiple levels of abstraction with consistency. However, some challenges and other viewpoints were also observed necessary while implementing it on real world case studies which

requires further research work in the developed frameworks as will be presented in Chapter 7.

6.4.5 Results via empirical evaluation

6.4.5.1 Empirical study setup

The developed IAT and CM in an integrated framework have been used by automotive engineers on live three different projects belonging to different levels of vehicle hierarchy (i.e. from feature to system to subsystem levels). The study scope of each of these projects varied from requirements elicitation to system architecture analysis across multiple levels of abstraction as summarised in Table 6.1. The results obtained only on IAT and CM within the integrated framework are discussed in this research.

Table 6.1 Overview of conducted empirical study

Empirical Study Setup		**Team 1 Project**	**Team 2 Project**	**Team 3 Project**
Vehicle Hierarchy	**Feature Level**	Electrical & control features		
	System Level		Braking System	
	Subsystem Level			Deployable rear active spoiler system
Project Study Scope		• Requirements Derivation	• Requirements Derivation & • System Architecture Analysis	• Requirements Derivation & • System Architecture Analysis

The team structure and scope of each case study within the vehicle hierarchy is summarised as follows:

- **Feature level within vehicle hierarchy**: The Team 1 from vehicle engineering discipline, based on two members, applied only the IAT on a number of new electrical and control features that mostly belong to software and electrical systems. The Team 1 aimed at deriving features' requirements via IAT.

- **System level within vehicle hierarchy**: The Team 2 from chassis engineering discipline, based on two members, (in which one member was just for review purpose) applied both IAT and CM on the braking system across its multiple levels of abstraction as discussed in Chapter 6.
- **Subsystem level within vehicle hierarchy:** The Team 3, based on three members, from body engineering applied both IAT and CM on active rear spoiler case study at one level of decomposition.

The team members were quite experienced and well familiar with the already existing tools available in industry such as interface analysis sheet/table and function tree etc. The empirical study process took 4 to 8 months' time period with meetings held once or twice a week.

6.4.5.2 Data Collection

Data were gathered in three different ways: documentation review, author participation, and interviews/workshops in person with team members for getting their opinions on IAT and CM usage. The data collection process was majorly qualitative. According to Seaman (1999), qualitative data is much richer in information than quantitative.

- During *documentation review*, the generated information was inspected to observe possible confusions, error and misunderstandings in the usage of developed IAT and CM.
- In *researcher participation*, the author played a role of assisting the technical and systems engineers applying the developed IAT and CM. This involved researcher answering questions to team members in person or via phone and email.
- Semi-structured interviews and workshops held during the study with team members to get the overview and feedback on the usefulness of the IAT and CM and on possible merits and demerits and risks with them.

6.4.5.3 Data Analysis

The collected data was categorised into following two stages by researcher:

- Assessing the differences in how the IAT and CM were applied on different case studies.
- Assessing the views and feedback responses of three teams via in person meetings with software, electrical, electro-mechanical, mechanical and systems engineering background.

6.4.5.4 Summary of obtained results

The results of the empirical study obtained through each team are now discussed in this section. Documentation review revealed that Team 2 and 3 applied the IAT and CM in complete formalism in contrast to Team 1.

Feature level Team 1: Team 1 applied IAT at feature level with incomplete formalism in its structure. The team members realised the effectiveness of the IAT in a way that it helped in highlighting the new requirements. The discovered requirements were escaped through existing requirements definition process. The reason being that existing requirements elicitation process focused on 'interaction operations' viewpoint. This viewpoint combination with 'exchange viewpoint' provided deeper understanding of interactions and thus the need of considering these two viewpoints together was recognised and appreciated by team members from IAT. They also used IAT in conjunction with existing requirements definition process that resulted in information overlapping. The reason was obvious as IAT combines diverse viewpoints available within use case and interface modelling methods. The IAT was not applied in isolation rather used in conjunction with existing requirements process that caused more time to team. They did not consider the exchange effect viewpoint on vehicle attributes which was also the case with existing requirements definition process. According to them, it complicates the process and adds no value. The rest of the viewpoints of IAT were applied consistently. Team 1 also suggested to revise the structure of IAT possibly to skip exchange effect and attributes consideration viewpoints.

System Level Team 2: Team 2 applied both IAT and CM with rigid formalism. Team 2 also found the effectiveness of IAT as it helped in identifying additional requirements at interfaces that were not captured with previous requirements documentation process. Team members also found that IAT provides much richer information both from operations and structural constraints perspectives

in contrast to existing interface analysis tables that mostly capture mechanical and structural interactions. Team 2 also stated that it helps in “describing what the system should do rather than what the identified solution does”. Team 2 members gave feedback that the structure of IAT is fine.

Team 2 also applied CM and recognised its benefit of coupling the interaction requirements to transformative functions and then transformative functions to subsystems. Team 2 specifically concluded that CM helps to “identify inconsistencies” between requirements and functions. Team also gave feedback that it takes time to fill each cell in the matrix and thus database or software prototype would be good to have formal links. On the other hand, it was also appreciated that CM gives deeper understanding of a system analysis. They reported it supports in creating and analysing multiple physical architectures with same set of allocated interaction requirements and transformative functions. Team 2 also gave constructive feedback that CM had reached its limit from functions’ decomposition perspective which requires further research.

Team 2 was quite encouraged and satisfied with the application of IAT and CM across multiple levels of abstraction. Team members gave very positive feedback that it was quite useful to have this top-down requirements cascading and repetition i.e. recursive aspect of an integrated framework. Team also recognised that the developed integrated architecture analysis framework underpinning IAT and CM promotes strong formalism of well-defined interfaces at multiple levels of abstraction and demands for establishing well braking system-of-systems hierarchy.

Sub-system Level Team 3: Team 3 also appreciated the need of IAT and CM in their current practice. Team 3 gave constructive feedback that performance requirements can vary from use case to use case which was considered and the IAT model was revised. Team 3 also recognised that capturing pair wise interaction functional requirements between two subsystems makes more clarity which is not very well tackled by already existing interface analysis table in the industry. However, Team 3 had problems in terms of analysing and managing the multiple ‘operating modes’ of active rear spoiler. This demanded for more structured functional modelling tool which was beyond the scope of IAT and

CM. Though use case diagram served the purpose but interdependencies analysis between multiple modes was a critical issue to manage. Thus, in future, it would be valuable to incorporate ‘modes of operation’ viewpoint into the integrated framework.

The obtained results through data analysis are summarised in Figure 6.25. The semi-structured interviews and opinions of multidisciplinary engineers in workshops showed that a majority of the members considered the developed IAT and CM useful as shown in Figure 6.25a. Figure 6.25b shows that how formally these were applied on case studies.

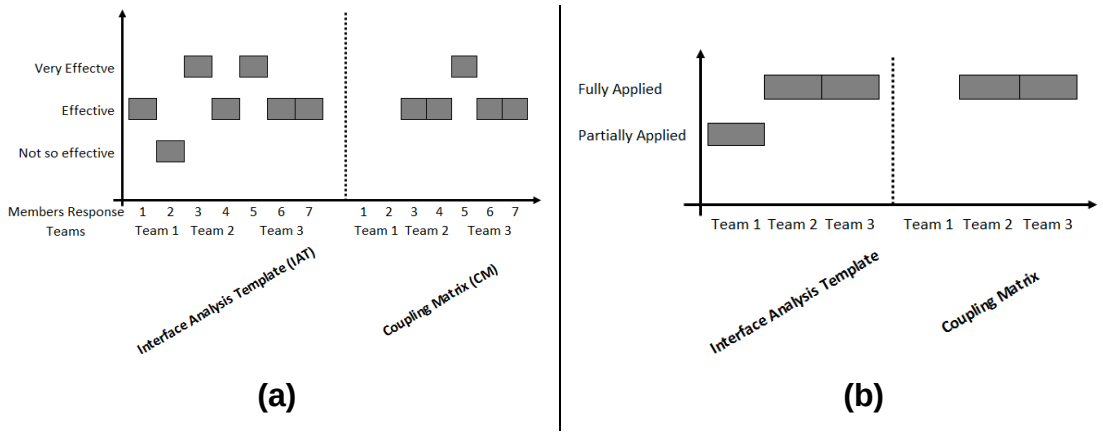


Figure 6.25 Effectiveness of developed IAT and CM

6.4.5.5 Relating results to research hypothesis

All members in the team appreciated the strengths of IAT and CM and recognised the needs of these two in the current practice. Above all, the developed IAT tool and CM framework proved their effectiveness for the purposes they designed for as follows:

- requirements derivation via integration of diverse interaction viewpoints at interfaces (i.e. *Research Hypothesis 1*: the IAT tool);
- and system architecture analysis in conjunction with well-defined interfaces (i.e. *Research Hypothesis 2:* the CM framework);
- The integrated architecture analysis framework (an architecting approach) (from section 6.4.1.1 to 6.4.1.7) proved to be highly effective across multiple levels of abstraction with a strong focus of well-defined interfaces (i.e. *Research Hypothesis 3*: Iterative and recursive aspects of IAT & CM across multiple hierarchical levels).

However, teams also noted that effectiveness achieved at the cost of time taking methodologies. There are mainly three key reasons behind this as observed by the author. First, the developed CM and IAT were new to automotive engineers and the engineers dealt with many interconnected viewpoints in these two for the first time. Secondly, the used language was another confusing factor to engineers as the notions used in IAT and CM were different from the notions and language that practiced by automotive engineers in industry. Lastly, the engineers' familiarity, technical knowledge and experience on systems engineering tools was also another key factor. It was particularly observed that Team 2's technical specialist had sound knowledge and experience on systems engineering process and modelling tools in contrast to other two teams. As a result of which Team 2 took twice the less time and delivered the thorough work up to two levels of abstraction with strict formalism which was the other way round in the other two cases where author's mentoring and assistance was needed more. This concludes that the developed IAT and CM require intensive coaching and training to automotive engineers along with incorporation of notions and technical language in the design methodologies which is adopted in industry.

6.5 Chapter summary

This chapter has presented the integrated framework underpinning IAT and CM in the context of systems engineering process which is validated on a real world engineering case study. The overall aim of this chapter has been to validate the integrated architecture analysis framework on a complex system real world engineering case having multiple levels of abstraction (or decomposition). The gained insights, observations and conclusions from a validation case study are discussed. The chapter then reflects on the practical usefulness of IAT and CM by using the results obtained during a conduct of empirical study in automotive industry with a set of independent engineers. The next chapter discusses the theoretical usefulness and strengths of the IAT, CM, and the integrated framework.

7. Discussion

7.1 Introduction

This chapter reflects on the use of developed IAT, CM and the integrated framework. The chapter initially discusses the iteration and recursive aspects of integrated architecture analysis framework using the reference model (Figure 3.1) and the content of Chapter 6. After this, the proposed IAT and CM are critically reviewed individually against the existing adjacent frameworks in academic literature within the matching fields. The practical usefulness of IAT and CM is also discussed in the context of FMA framework suitable for automotive industry practice.

7.2 The key points of the research

This research has revealed following key points:

- There is no concrete interface definition methodology or model (as summarised in Chapter 3) for deriving system requirements that could be applied consistently both across solution independent (black-box) and dependent analysis (white-box); IAT has been developed and validated in Chapters 4 to 6.
- There is no detail architecture analysis model that could promote system-of-systems thinking across multiple levels of abstraction (as discussed in Chapter 3) with consistent definition of multiple views and traceability in between them in conjunction with well-defined interfaces with intensive information for which CM has been developed. The CM framework validated at one level of decomposition in Chapter 5 and showed a need of recursive aspect of its deployment which subsequently conducted in Chapter 6.
- The IAT and CM have been encapsulated in an integrated architecture analysis framework with other supporting tools for validation purpose at multiple levels of abstraction through an industrial case study that revealed promising results to real world engineers as seen in Chapter 6.

Section 7.3 discusses the integrated architecture analysis framework from iteration and recursive aspects using the reference model thereby relating its usage to multiple levels of systems hierarchy.

7.3 The reference model for system architecture analysis

7.3.1 The integrated framework in the reference model

It is seen in this research that there is no concrete reference model for system architecture analysis that could conceptualise basic necessary views, their types and viewpoints in it along with highly iterative (between views) and recursive aspects (between levels) as discussed in Chapter 3. This research establishes the reference model (Figure 3.1), that helped in deriving IAT tool and CM framework suitable for iterative and recursive aspects.

The integrated framework iterative aspect: The 'iterations' can occur across ***multiple views*** (e.g. requirements, functions, and structure views) when reference architectural model is applied to a system-of-interest at only one level of decomposition (i.e. from black-box to white-box). As discussed in Figure 6.1, the requirement view (interaction requirements) is linked to functional view (transformative functions) and subsequently to design view (internal actors/subsystems) which can be achieved by the integrated framework underpinning IAT and CM. The CM helps in managing the iterations from requirement to physical design views as summarised with clockwise arrow-head in Figure 7.1. The IAT double arrow-head in Figure 7.1 shows the system-of-interest interfaces identification with external actors and then thereafter deriving its requirements.

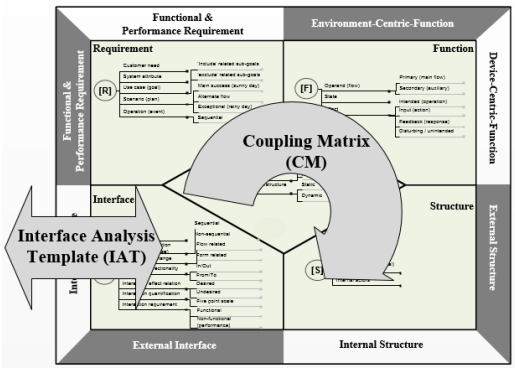


Figure 7.1 The integrated framework's iteration aspect at one level of decomposition

For example, as an evidence in Chapter 6, the regenerative braking system requirements are modelled at system's black-box via IAT (Figure 6.5) whereas CM framework connects the requirements to transformative functions to internal solutions at white-box (Figure 6.9). There can be highly iterative feedback loops in between them such as requirements loop between requirement and function analysis views (Figure 6.7). Similarly, the design loop between function and (internal) structure views (Figure 6.9).

The integrated framework recursive aspect: The 'recursions' can occur across *multiple levels of abstraction* (e.g. system, subsystem, and component levels) when reference architectural model is applied to every system-of-interest existing within another system-of-interest hierarchy (i.e. system-of-systems). As discussed in Chapter 6, the actuation system exists within a brake system's hierarchy at level-1 between level-0 to level-2. This is illustrated graphically, in Figure 7.2, with the reference model applied recursively in a design hierarchy.

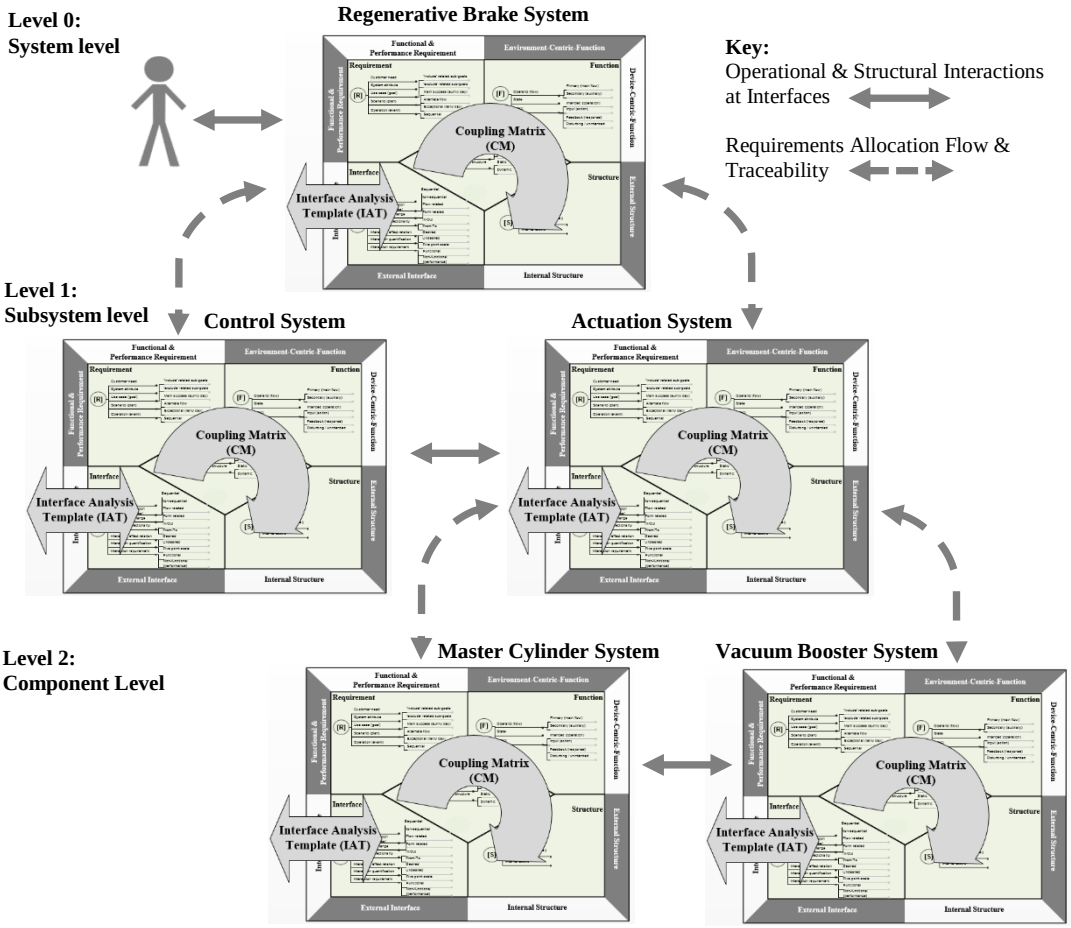


Figure 7.2 The integrated framework's recursive aspect at multiple levels of abstraction/decomposition

The key point to note is that next level (e.g. subsystem level) knowledge is built upon the previous knowledge available in above level (e.g. system level) both from requirements flowing from top level as well as interaction requirements from same level. For example, the interaction requirements knowledge in IAT of regenerative braking system at top level-0 interfaces (Figure 6.5) and at level-1 (Figure 6.13) interfaces became the external interaction requirements knowledge for next system-of-interest i.e. the actuation system at level-1 (Figure 6.15) as illustrated graphically using the reference model in Figure 7.2. The actuation system's requirements are then coupled with its own transformative functions via CM and subsequently allocated down to its components such as vacuum booster at level-2. The key point to note is that the integrated framework was applicable recursively to each level of hierarchy. The Chapter 6 in this research has clearly shown both the iterative and recursive aspects of integrated architecture analysis framework based on IAT and CM.

7.3.2 The scope and procedure of integrated framework underpinning IAT and CM

Figure 7.3 shows the scope and the procedure of proposed integrated framework based on IAT and CM in the interface and architecture-cube-models (Figure 3.3).

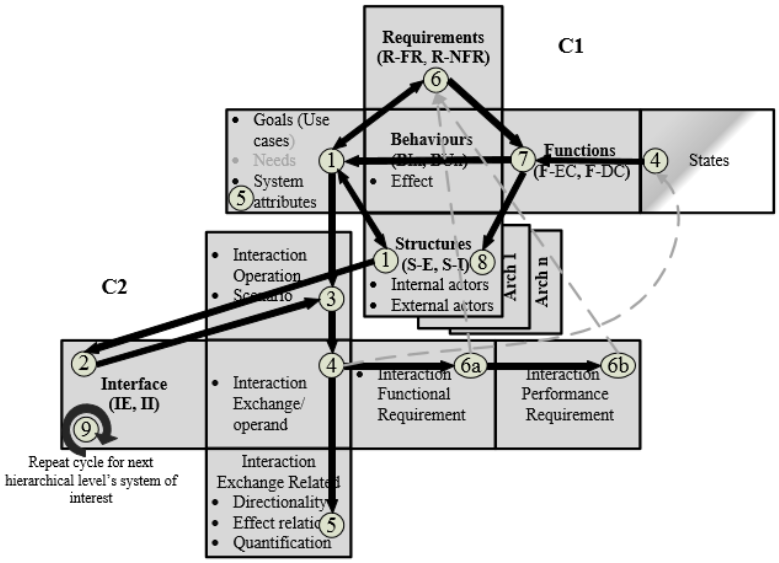


Figure 7.3 Integrated architecture analysis framework scope and procedure

Figure 7.3 shows that integrated framework begins with the context definition thereby identifying system goals and linking to external actors in the step 1,

followed by interface definition between external actors and the system in step 2. The interaction operations and exchanges of system with its external interfaces are identified in step 3 and 4 respectively. The exchange impact is analysed on system's engineering attributes in step 5 and using the step 3 information, technical interaction (both function and non-functional) requirements are derived in step 6. The transformative functions are identified in step 7 using the main exchanges identified in step 4 which are then coupled with requirements (of step 6) and then allocated to internal subsystems in step 8. The interface analysis are again conducted for chosen architecture's internal actors/subsystems in step 8. The clockwise directed circle at the bottom in the cube 2 (with step 9) shows the repetition cycle aspect for the analysis of next hierarchical level's system-of-interest. Figure 7.3 can also be used to show the individualised scope and procedure of CM and IAT. For example, in Figure 7.3, the interface cube-model C2 shows the scope and procedure of IAT. The architecture cube-model C1 shows the scope and procedure of CM.

7.4 The proposed IAT and CM and other existing approaches

In this section, the developed IAT tool and CM framework are contrasted with the other strengths of existing approaches that were beyond the defined scope of this research with the intent to identify any potential conflicts and potentialities for other benefits from developed IAT and CM.

7.4.1 The CM and adjacent approaches

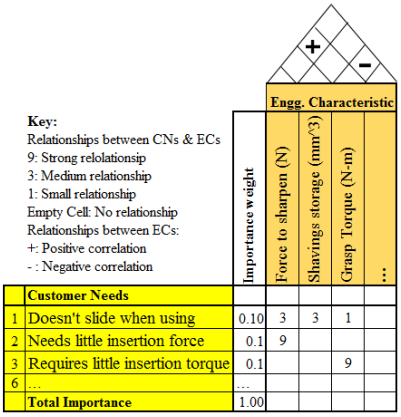
In this section, it is explored in which specific way the CM may be used and adapted, in order to support other existing system architecting modelling approaches.

Quality function deployment: In quality function deployment, customer needs are articulated in unstructured manner and seems to be a combination of two viewpoints: 'use cases' and 'interaction operations' viewpoints. The CM, on the other hand, clearly structures these two viewpoints separately as shown in Figure 7.4 via an example based on the electric pencil sharpener used in Chapter 4. Quality function deployment transforms customer needs into engineering requirements without discussing the system interactions and functions in detail as it mainly captures end user interactions and avoids

capturing other operating and enabling systems interactions that may lead to incomplete set of technical requirements on a system. The complex system can exist in any hierarchy level that interacts with other enabling systems and other environmental stakeholders that often affect the system which CM framework manages such transition from uses cases till engineering interaction requirements via IAT and then linking those to transformative functions as shown in Figure 7.4.

The other key strength of quality function deployment is that it shows 'interrelated dependencies' using correlation matrix with 'positive/negative (+/-)' relationships between engineering requirements which CM can cover using functions definition. For example, shown in Figure 7.4a 'torque' is dependent on 'force to sharpen' with positive (+) relationship which CM can also show that function 'cut/separate pencil' encapsulates both these non-functional performance requirements. This reveals the mathematical *transitive* property that if 'torque' is required for transformative function 'cut pencil' and also 'force to sharpen' is required by same function then it is obvious that these two requirements are dependent and can influence/impact each other and thus 'positive/negative' relationships as shown in Figure 7.4b.

Hence, in addition to various viewpoints coverage, the proposed CM can also provide other additional benefits underpinning the quality function deployment.



	Customer Needs	Importance weight	Engg. Characteristic: Force to sharpen (N)	Shavings storage (mm^3)	Grasp Torque (N-m)	...
1	Doesn't slide when using	0.10	3	3	1	
2	Needs little insertion force	0.1	9			
3	Requires little insertion torque	0.1			9	
6	...	...				
	Total Importance	1.00				

(a) Quality function deployment

Use Cases	UC1: Transport sharpener			×
	UC2: Provide sharp lead pencil	×	×	...

Interface Analysis Template				Transformative Functions		
C1: Interface	C3: Interaction Description	C11: Interaction Functional Requirements	C12: Non-functional performance requirements	F1 Load Pencil	F2 Cut Pencil	F3 ...
User – Sharpening System	Sharpening system requires user input	Import Human Force	Force to Sharpen (X< N<Y)	X	+	...
	Sharpening system slides from user during sharpening operation	Reduce Sliding	Surface friction (X<u<Y)		-	
Pencil – Sharpening system	Sharpening system removes pencil material	Transmit Cutting Energy	Cutting Energy (N/m^2)		-	...
	Sharpening system requires pencil insertion torque	Grasp Torque	Torque (X< N-m <Y)		+	...

(b) Coupling matrix

Figure 7.4 Mapping of interdependencies between technical requirements

Funkey coupling matrix: Bonnema (2008) provides FunKey coupling matrix where stakeholders' key drivers/performance aspects on a system are coupled

with ‘functions’. The developed CM also links functions with performance requirements. The Funkey also aids designers to think of ‘budgets requirements distribution’ (Table 2.4) across system functions and subsystems which CM framework can also benefit as shown in Figure 7.5.

Figure 7.5a shows that budgets (target values) of performance requirements have been distributed to various functions of the vehicle system. For example, the overall budgets of cost and power are split into sub-budgets and then distributed/allocated to the functions that should contribute for overall budgets. Similarly, in CM in Figure 7.5b, the electric energy and insertion force related overall budgets can be distributed on transformative functions.

Function	cost	power	visibility	effort	subsystem
drive vehicle					
– deliver force				*	
– transmit force	20				Drive and frame
maintain balance	20				Drive and frame
maintain posture	20				Drive and frame
create light					
– on road	5	40	60	1	Electrical
– to other road users	5		40		Drive and frame
steer				*	
navigate					Navigation

(a) Funkey (adapted from Bonnema, 2008)

Use Cases	UC1: Transport sharpener			×
	UC2: Provide sharp lead pencil	×	×	...

Interface Analysis Template				Transformative Functions			
C1: Interface	C3: Interaction Description	C11: Interaction Functional Requirements	C12: Non-functional performance requirements	F1 Load Pencil	F2 Cut Pencil	F3 ...	
User – Sharpening System	Sharpening system requires user input	Import Human Force	Force to Sharpen (X< N<Y)	5	10	...	
Pencil – Sharpening system	Sharpening system removes pencil material	Transmit Cutting Energy	Cutting Energy (N/m²)		...	...	
	Sharpening system requires insertion torque	Grasp Torque	Torque (X< N-m <Y)		7	...	
Power Source –Sharpening System	Sharpening system requires electric power to operate	Import Electric Energy	Electric voltage (120 Volts)		70	50	Technical Subsystems
				X			Pencil Fixture
					X		Pencil Cutter

(b) Coupling Matrix

Figure 7.5 Distribution of budgets between functions

Integrated function modelling approach: Eisenbart (2014) provides integrated function modelling framework where each matrix connects two different viewpoints but with a central viewpoint of ‘technical process’. The presented CM framework has ‘transformative functions’ as a central viewpoint in each matrix connected to other views. The integrated function modelling framework, on one hand, does not discuss the connectivity of requirements view which is covered by the proposed CM framework via its integration with IAT. On the other hand, the key strength of integrated function modelling framework is its ability to show the states of both ‘exchanges/operands’ and ‘technical subsystems’ in a separate matrix. The CM framework does not have a separate matrix for states representation rather uses functional modelling tool to show intermediate states of ‘exchanges/operands’ only. The CM shows input/output states of exchanges/operands only via IAT (Figure 5.3 and 5.4) and

the room is still there to explore the aspect of modelling the ‘subsystem’s states’ too into the CM as represented in Figure 7.3.

This section has clearly pointed out that existing system architecting frameworks lack to show the role of interface definition which is quite important from requirements derivation and system-of-systems thinking perspective and thus an architecting method should incorporate and integrate interface view too.

7.4.1.1 Implications

The CM have met the requirements outlined in Section 3.6.2.1 in Chapter 3. In addition to that the comparison with the existing approaches in previous section (Figures 7.5 and 7.6) has also revealed that CM can be adapted and can offer more. The previous section also ensures and concludes that CM is covering the key views of various existing approaches designed for system architecting. Furthermore, there is still a room for further research in CM framework in order to explore the relationships with other approaches in order to get full benefits from it.

Matrix-based models such as quality function deployment, and design structure matrix are widely applied in engineering practice (Eckert, 2013; Lopez-Mesa & Bylund, 2011). These sort of approaches often generate large matrices but still are widely applied in design practice (Lopez-Mesa & Bylund, 2011). Thus, same can be expected from the implementation perspective of CM framework.

In order to address the issues of complexity and modelling efforts among various views and viewpoints, one potential support that is required, is the software tool or prototype for implementing the CM. The CM framework in current shape is document based which takes time like other documents driven matrix based approaches.

7.4.2 The IAT and adjacent approaches

In the following section, it is explored in which specific way the IAT may be used and adapted in order to support other use case and interface modelling approaches dealing with derivation of requirements. For the purpose of this analysis, the different viewpoints in the IAT are mapped onto the existing interface definition and use case modelling methods.

Daniels & Bahill's hybrid model: Daniels & Bahill's (2004) approach with related modelling tools are shown in Table 7.1. The first modelling step of proposed IAT is similar to Denis & Bahill's approach i.e. defining use cases of a system related to its external actors. The columns C1 and C13 of IAT along with use case diagram is used for this purpose. In the next step of Daniels & Bahill's method, sequence of events/operations are defined which IAT covers in column C3. In the final step of Daniels & Bahill's method, both functional and non-functional requirements are derived based on step 2. The IAT's column C11 and C12 can be used in similar way by skipping exchanges and attributes related viewpoints. There are other sub-viewpoints associated with interaction operations that are often suggested by Daniels & Bahill named as 'rules' and 'pre and post conditions'. It would be interesting to explore the relationship of IAT viewpoints with these viewpoints.

Table 7.1 The IAT and Daniels & Bahill (2004) use case model

Step	Description	Related modelling tools	IAT
1	Identifying use cases and external actors	Use case diagram	Use case diagram, IAT – C13
2	Specify sequence of interaction operations (events) for a use case in scenarios	Use case standard template	IAT – C3
3	Identifying functional requirements from use case events and rules	Traditional shall statements	IAT – C11
4	Identifying Non-functional requirements from use case's events and rules	Traditional shall statements	IAT – C12

Weilkiens's use case realisation and interaction model: Weilkiens (2007) methodological steps are shown in Table 7.2.The IAT is capable of achieving same steps in respective columns as mapped. Table 7.2 shows that Weilkiens uses first two steps (i.e. use cases and system context) as an inputs for modelling system/actor interactions which can be modelled via IAT's columns C1, C2, C5, and C13. Weilkiens realises a 'use case' by beginning with system/actor 'interaction operations (as activities)' via interaction/sequence diagram and then derives 'interaction points (to him interfaces) as exchanges' from system/actor interaction model onto the block-definition diagram. There is no clear difference drawn between 'interaction operation' and 'interaction functional requirements', thus it can assumed that Weilkiens' interaction model

represents either interaction operations or functional requirement viewpoint which can be modelled in IAT in any of respective columns as mapped in the Table 7.2.

Weilkiens shows that only one exchange as an interaction point can be derived from each interaction modelled between external actor and system whereas IAT has shown and argued that there can be one or many exchanges in an interaction operation (Figure 4.15). Note that IAT promote voice of customer language in interaction operation whereas it is often used with technical description (i.e. voice of engineer) as observed in existing sequence diagrams or see interaction descriptions of interaction model in (Weilkiens, 2008). Furthermore, Weilkiens discusses the characterisation of system's exchanges/interaction points from ports perspective: required (from actor) and provided (to actor) in block-definition diagram, which can be achieved in IAT in the 'from (actor)' or 'to (actor)' in columns (C7-C8). Note that Weilkiens does not discuss explicitly how to derive system's functional and non-functional requirements from use case modelling and interface modelling viewpoints rather he shows inversely that system actors and system context are derived from general requirements.

Table 7.2 The IAT and Weilkiens use case realisation model

Step	Description	Related modelling tools	IAT
1	Identifying use cases of a system with external stakeholders and other technical systems	Use case diagram	Use Case Diagram, IAT – C13
2	Identifying system context thereby identifying its possible flows with external stakeholders and enabling systems	System context diagram	System context diagram, IAT – C1, C2, C5
3	Modelling system/actor interactions as processes or operations within a use case	Interaction/ sequence diagram	IAT – C3 or C11
4	Deriving system interfaces (as interaction points)	Block definition diagram	IAT – C5, C7, C8

Fritzsche's interface description model: The interface modelling method proposed by Fritzsche (2008) is mainly a representation of an automotive industrial practice which is applied at subsystems or components level. The Fritzsche approach can be mapped onto IAT to a large extent as shown in Table 7.3. While the Fritzsche interface analysis approach covers many common aspects as available in IAT, it does not discuss the role of linkage of

‘interaction functional requirements’ with either the ‘use cases’ or ‘transformative functions’. It assesses the exchange importance on system’s interaction functionality. Furthermore, interaction functional descriptions are derived based on performance targets whereas in IAT it is other way round.

The other aspects in Fritzsche model are also important such as IAT tool can be adapted to incorporate the aspects of ‘exchange importance’ and ‘exchange effect’ separately. Fritzsche also bring the role of people responsible (i.e. an interface partner) for maintaining or managing an interface which again IAT can be benefited from thereby adding another columns and thus adapting it.

Table 7.3 The IAT and Fritzsche’s interface description model

Step	Description	Related columns in Interface description sheet	IAT
1	Identify interfaces	Interface Matrix, C3, C4	IAT – C2
3	Identifying interactions as exchanges		IAT – C4, C5
4	Specifying all functional targets	C5, C6,	IAT – C6 & C12
5	Technical functional description	C7, C8	IAT – C11
6	Identifying exchange importance on function	C9	
7	Identifying exchange impact	C10	IAT – C10
8	Identifying interface partner	C11	

In this section, it can clearly be pointed out that existing use case and interface modelling methods lack to show their linkage with other system architecting activities i.e. requirements linkage with functions and their allocations to multiple architectures’ subsystems.

7.4.2.1 Implications

The IAT have met the requirements outlined in Section 3.6.1.1 in Chapter 3. In addition to that, previous section has also revealed that the IAT is compatible with the existing use case and interface modelling approaches.

The comparison highlighted the facts that how the specific contents or viewpoints of existing interaction modelling approaches, may be represented using the viewpoints integrated in the IAT. The discussed methodological steps of both IAT and other modelling approaches have also been mapped which clearly show that IAT does not by pass any interaction modelling viewpoint like

other interaction modelling approaches and covers all necessary viewpoints (Tables 7.1 to 7.3). There are some other aspects that IAT can benefit from with some adaptations as observed and discussed in Daniels & Bahill and Fritzsche's modelling approaches.

A potential weakness of the IAT at the moment is that it can generate long documents for relatively complex systems due to integration of diverse interaction viewpoints. Also it is document driven approach that takes time and thus requires a relational database or software package to organise and manage the IAT model. The important point to note is to look for the output of IAT and its guidelines on increasing of detail between various viewpoints with other modelling methods. For example, Weilkiens (2006) uses SysML based diagrams such as use case, context and sequence and block definition diagrams whilst Daniels & Bahill (2004) use case diagram and textual template to model various viewpoints of interactions. On the other hand, IAT mainly uses tabular template in conjunction with use case and context diagrams. From the output and contents coverage perspective, IAT stands out stronger due to richer knowledge in one template whilst on the other hand it can also be heavy due to lack of graphical structure as seen in SysML tools. The other advantage with IAT's tabular template (and a key reason behind) is that as mostly researchers and engineers in academia and industry understand and use tabular templates in contrast to relatively new modelling languages such as SysML based tools.

7.5 The practical role and use of IAT and CM in industrial context

7.5.1 FMA framework: The current practice in industry

The BEQIC research has developed an integrated FMA framework, in Figure 7.6, based on four key stages underpinned by a series of tools that are well aligned with the failure mode avoidance process practice in automotive industry (JLR, 2014; Ford, 2004). It ensures that coherent information flows between tools thereby providing strong function and failure modes reasoning. The integrated FMA framework promotes functions identification via system state flow diagram and then subsequently identifies interaction functional requirements for a chosen architecture via interface analysis table as shown in Figure 7.6. This framework has been adopted by automotive engineers and

proved to be very useful at the physical systems layers (i.e. from subsystems to component to manufacturing levels).

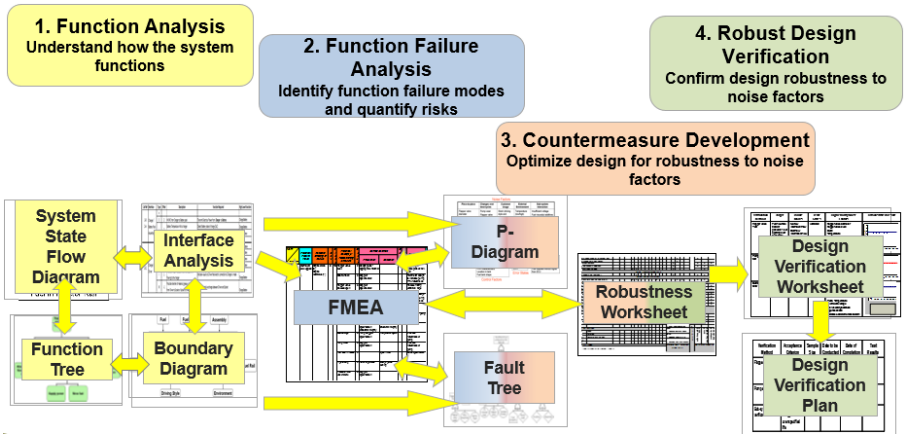


Figure 7.6 The BEQIC FMA framework (Campean et al., 2013)

The functional analysis stage of FMA framework is also well aligned with systems engineering principles. It also covers verification activities as can be seen in Figure 7.6. However, it is noted in this research that FMA framework is not fully integrated with the systems engineering design process' stages and activities due to following key reasons:

- 'Requirements analysis' stage in FMA framework is missing which means there is no supporting path or tool for deriving system requirements with respect to its services (as goals) and interaction operational analysis associated with external stakeholders'.
- Misalignment between stages of integrated FMA framework and systems engineering process. For example, the FMA framework's 'functional analysis stage seems to have sets of tools that belong to 'functional analysis' and 'design synthesis' stages of systems engineering process.
- In the functional analysis stage of integrated FMA framework, there is no traceability or allocation path for requirements loop that exist between 'requirements analysis' and 'functional analysis & allocation' stages. Also, there is no traceability or allocation path of design loop that iteratively occurs between 'functional analysis & allocation' and 'design synthesis' stages as recommended by systems engineering process.

The present research has managed to fill these gaps. The IAT is proposed, on one hand, for deriving requirements in requirements analysis stage at feature

and system levels. On the other hand, the CM framework is designed for establishing traceability and requirements allocation paths from requirements analysis to functional analysis and allocation till design synthesis stage in FMA framework as per systems engineering process as shown in Figure 7.7 (see also Figure 6.3).

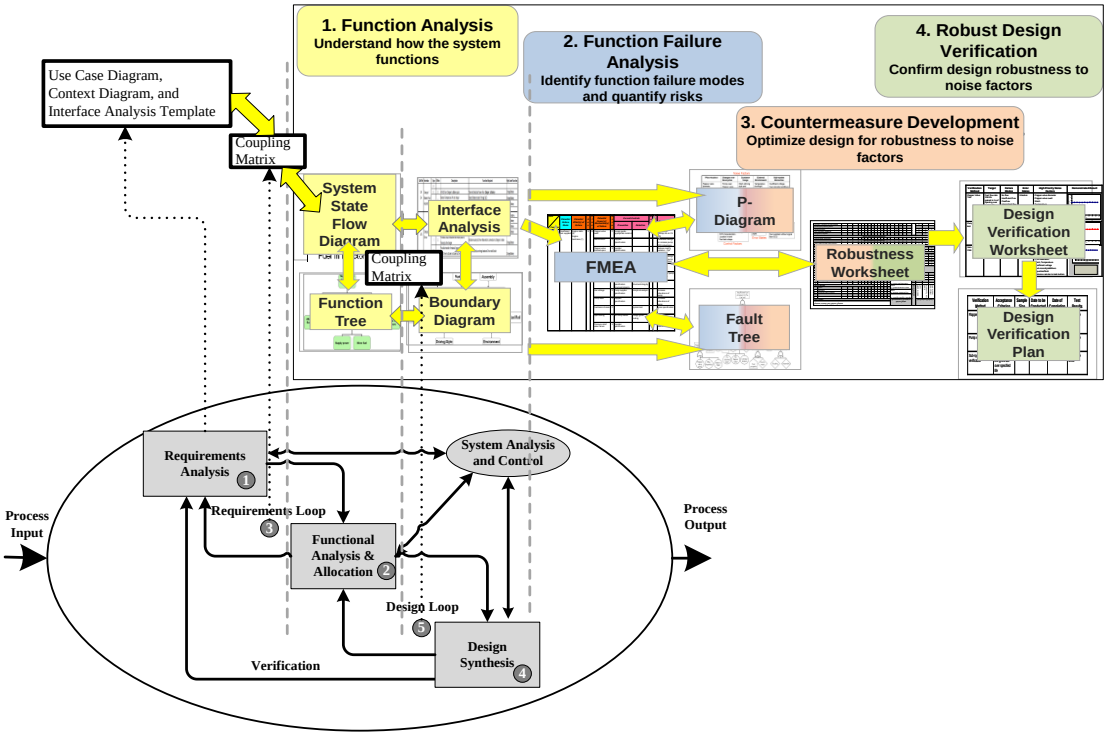


Figure 7.7 The research contribution by integrating systems engineering process activities in the FMA framework via IAT and CM

The IAT and CM have been validated within an automotive company with a set of independent engineers with overall positive feedback (Section 6.4.5).

7.6 Chapter summary

This chapter has initially discussed the theoretical usefulness of developed integrated framework underpinning IAT and CM by using the reference model from the perspectives of iteration and recursive aspects as well as the scope and procedure of it. The proposed IAT and CM are also contrasted with other existing approaches and it is highlighted that there is more to offer by these two with possible adaptations and extensions. This requires further research work. The next chapter summarises the key contributions and conclusions of the research with recommendations for future work.

8. Conclusions and Recommendations for Future Work

8.1 Introduction

This chapter reviews and presents the key contributions of this research, followed by conclusions and recommendation for future work.

Review of research objectives:

8.2 Review of research contributions

The research contributions in the light of research objectives are summarised in this section as follows:

- The first objective of this research was to carry out systematic literature review (i.e. Objective 1 in Chapter 1). This thesis has carried out systematic literature review based on initial grouping of system architecture definitions that helped in extracting five key essential modelling views and also in deriving **three taxonomies** for system modelling and solution development: (i) viewpoints in each views, (ii) types of each view and (iii) system decomposition views on a hierarchy (Chapter 2).
- The second objective of this research was to evaluate the existing approaches in system interface and architectural analysis via mutually exclusive and collectively exhaustive criteria (i.e. Objective 2 in Chapter 1). The three taxonomies were integrated into a reference model. Thus, this thesis has **developed a reference model** comprising of key views and viewpoints of system architecture that helped in extracting the mutually exclusive and collectively exhaustive criteria for comparing the scope and procedure of existing approaches. The reference model integrates the essential viewpoints and types across system's black-box and white-box levels that are used for the definition and analysis of each modelling view. This sort of detail reference model that could accumulate system architecture modelling views, viewpoints and types across one level of decomposition is not available in systems engineering and engineering design communities (Chapter 3).

- The third objective was to develop an integrated approach centred on interface modelling methodology at one abstraction level (i.e. Objective 3 of Chapter 1). To accomplish this objective, the thesis has developed first an **original interface definition methodology** in the form of **IAT tool** via existing case study and concepts' reasoning around it, for interface analysis and requirements derivation. The IAT integrates the key modelling diverse viewpoints for system interface analysis. The iterative aspect (two iterative loops: loop 1 & 2) within IAT at requirements analysis stage is also presented (Chapter 4).
- The **IAT's UML model** was introduced with a view that interface definition methodology or model can be transformed into any software prototype format: tabular or graphical, but emphasis is on final output in the form of IAT structure that should involve input modelling elements aggregated into IAT UML model (Chapter 4).
- The third objective (i.e. Objective 3 in Chapter 1) of this research was further perceived with the emphasis of integrating the other system architecture views with the methodology centred on interface modelling at one level of decomposition. The reasoning of IAT via ball pen case study with other architectural views led to CM development. The thesis has contributed by extending the scope of available coupling matrices used for definition and creation of system architecture thereby introducing **original CM framework** that draws clear distinction between 'use cases', 'interaction functional requirements', '(non-functional) performance requirements', 'transformative functions' and their integration and allocation to 'subsystems' in an architecture (Chapter 5).
- This thesis, for the first time, has contributed significantly by showing integration of interface view with other system architecture views thereby proposing **CM framework in conjunction with IAT tool** (Chapter 5).
- The **architecture analysis UML model** was also introduced after integrating IAT UML model with the CM framework viewpoints with a view that these elements should be covered by an architecting approach and the same UML model can be used to develop a software prototype (Chapter 5).

- The fourth objective of this research was to also validate the integrated architecting framework across multiple decomposition levels and real world engineering problems (i.e. Objective 4 in Chapter 1). The framework was validated through braking system example. The thesis has developed **integrated architecture analysis framework** based on novel IAT and CM with other existing modelling tools in the context of systems engineering process that can be applied ***iteratively across multiple views*** at one level and ***recursively across multiple levels of abstraction/decomposition*** of a complex system (Chapter 6). The existing approaches often lack to show iteration and recursive aspects from system-of-systems thinking perspective.
- The final objective of this research was to reflect on the results obtained while applying the developed framework on many desktop and real world engineering systems and relating them to research hypothesis (Objective 5 in Chapter 1). The thesis has contributed **to industrial practice** by validating the IAT and CM with a set of real world case studies and engineers of an automotive company **that revealed effective results** in their projects in the fields of interaction requirements and architecture analysis (Chapter 6).
- The usefulness of **IAT and CM beyond the thesis scope** with existing approaches' strengths were also discussed (Objective 5 in Chapter 1). It was shown that the IAT and CM have got more to offer and can possibly be extended in future research. The thesis also showed the contribution of IAT and CM to an **industrial practice** in the context of **FMA framework** (Chapter 7).

8.3 Conclusions

Based on the research theoretical and practical results, the following conclusions are drawn:

- The reference model can be used to assess the scope and procedure of existing modelling approaches in the fields of interface, requirements, and architecture analysis.
- The IAT provides intensive information and can be applied with same structure at black-box and white-box views of a system.

- The IAT helps in deriving system and interaction requirements more robustly in contrast to existing requirements derivation and ICDs documents.
- The CM couples the requirement, function, behaviour, structure, and interface views in a structured manner and provides intensive information than the existing architecting approaches.
- The integrated architecting framework based on IAT and CM supports complete and correct information traceability between multiple views at one level of hierarchy as well as consistent analysis of complex system across its multiple hierarchical or decomposition levels.
- The IAT and CM are effective enough practically as realised by a set of real world of independent engineers along with some difficulties due to notions and manual driven documents structures. Thus, it is concluded that IAT and CM do work and deserve to be further enhanced and expanded.

8.4 Recommendations for future work

This work has generated scope for further research in following aspects:

- First and foremost priority of this research now is to develop a software prototype that should be encapsulating IAT and CM viewpoints and thereafter should be evaluated further in real world industrial environment. It should cover following features:
    - It should allow establishing formal links between multiple views for traceability and allocation perspective;
    - It should allow zoom-in and zoom-out capabilities so that a relevant information of a system-of-interest in a relevant hierarchical level can be visualised and should be updated.
- In addition to that, following two viewpoints should also be incorporated into the developed integrated architecting framework: incorporation of 'multiple ***modes*** of operation' viewpoint with system's use cases definition as well as incorporation of '***states***' viewpoint for analysing states' transition for technical subsystems.

- The developed IAT and CM support left-side of V-model's activities and their linkage and integration with right-side of V-model activities such as verification and validation should be explored.
- This thesis mainly supports the management and analysis of expected behaviour of a system via IAT and CM. The re-iteration of design aspects and activities such as CAD and simulation modelling activities (i.e. verification and validation phases) that help in capturing system's actual behaviour should be integrated with developed integrated architecting framework which should allow updating and discarding the relevant generated information accordingly in IAT and CM.

References

Acharya, S. & Vidhi, P., 2012. Bridge between Black Box and White Box – Gray Box Testing Technique. *International Journal of Electronics and Computer Science Engineering*, 2(1), pp.175–185.

AEEC, 1997. *AEEC: ARINC-651: Design Guidance for Integrated Modular Avionic, Aeronautical Radio*,

Ahmed, S. & Wallace, K., 2003. Evaluating a Functional Basis. In *Proceedings of ASME, IDETC/CIE*.

Albers, A. et al., 2009. Contact and Channel Modelling to Support Early Design Principles. In *Proceedings of ICED*. 24-27 August, Stanford, USA.

Albers, A. & Matthiesen, S., 2002. Konstruktionsmethodisches Grundmodell zum Zusammenhang von Gestalt und Funktion technischer Systeme - Das Elementmodell ‚Wirkflächenpaare & Leitstützstrukturen' zur Analyse und Synthese technischer Systeme, Konstruktion, Zeitschrift für Produktentwicklung. , 54, pp.55–60.

Albers, A., Ohmer, M. & Eckert, C., 2004. Engineering Design in a Different Way : cognitive per- spective on the contact and channel model approach. In *Visual and Spatial Reasoning in Design*. 22-23 July, Cambridge, MA, USA: The Open University, pp. 1–21. Available at: oro.open.ac.uk.

Albers, A. & Zingel, C., 2011. Interdisciplinary Systems Modelling using the Contact & Channel-Model for SysML. In *International Conference on Engineering Design*.

Alexander, I. & Zink, T., 2002. Introduction to Systems Engineering with Use Cases. *Computing and Control Engineering Journal*, (December), pp.289–297.

American Heritage Dictionary, 2016. The American Heritage Dictionary. Available at: https://ahdictionary.com/.

Baldwin, C. & Clark, K., 2000. *Design Rules: The Power of Modularity*, Cambridge, MA: MIT Press.

Bartolomei, J.E., 2007. *Qualitative Knowledge Construction for Engineering Systems : Extending the Design Structure Matrix Methodology in Scope and Procedure*. PhD Thesis, MIT.

Bijan, Y. et al., 2013. Systems Requirements Engineering — State of the Methodology. *Systems Engineering*, 16(3).

Blackenfelt, M., 2000. Design of Robust Interfaces in Modular Products. In *ASME Design Engineering Technical Conferences*. Baltimore, Maryland, pp. 1–9.

Blackenfelt, M., 2001. *Managing Complexity by Product Modularisation*. Royal Institute of Technology, Stockholm, Sweden.

Blanchard, B.S. & Fabrycky, W.J., 1998. *Systems Engineering and Analysis* 3rd Edn., ed., New Jersey: Prentice Hall.

Blyler, J., 2004. Interface Management. *IEEE Instrumentation & Measurement Magazine*, pp.32–37.

Bonnema, G.M., 2008. *FunKey Architecting*. PhD Thesis, University of Twente Enschede, the Netherlands.

Bonnema, G.M., 2011. Insight , Innovation , and the Big Picture in System Design Application of FunKey Architecting. *Systems Engineering*, 14(3), pp.223–238.

Breu, R. et al., 1998. Systems, Views and Models of UML. In *The Unified Modeling Language, Technical Aspects and Applications*. Physica Verlag, Heidelberg, pp. 93–109.

Brown, D.C. & Blessing, L., 2005. The Relationship Between Function And Affordance. In *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. pp. 1–6.

Buede, D.M., 2009. *The Engineering Design of Systems: Models and Methods* 2nd ed., New Jersey: John Wiley & Sons, Inc.

Burge, S., 2007. A Functional Approach to Quality Function Deployment.

Burge, S., 2011. The Systems Engineering Tool Box. Available at: http://www.burgehugheswalsh.co.uk/.

Cabigiosu, A., Zirpoli, F. & Camuffo, A., 2013. Modularity, interfaces definition and the integration of external sources of innovation in the automotive industry. *Research Policy*, 42(3), pp.662–675.

Campean, F., Henshall, E. & Brunson, D., 2010. Faiure Mode Avoidance Paradigm in Automotive Engineering Design. In *CONAT20106025*. pp. 207–2014.

Campean, I.F. et al., 2011. A Structured Approach for Function Analysis of Complex Automotive Systems. *SAE Int. J. Mater. Manuf.*, 4(1).

Campean, I.F. et al., 2013. A Structured Approach for Function Based Decomposition of Complex Multi-discipinary Systems. In *Proceedings of the 23th CIRP design conference*. Bochum, Germany, pp. 113–123.

Campean, I.F. & Henshall, E.J., 2012. The Functional Basis for Failure Mode Avoidance in Automotive Systems Engineering Design. In *2nd International conference on modelling and management of engineering processes (MMEP)*. Cambridge, UK.

Cao, D. et al., 2013. Port-Based Ontology Modeling for Robot Leg Conceptual Design. *Advances in Mechanical Engineering*, 2013, pp.1–10. Available at: http://www.hindawi.com/journals/ame/2013/657960/.

Castaneda, V. et al., 2010. The Use of Ontologies in Requirements Engineering. *Global Journal of Researches in Engineering*, (September 2016).

Chandrasekaran, B., 2005. Representing Function: Relating Functional Representation and Functional Modeling Research Streams. *Artificial Intelligence for Engineering Design*, 19(2), pp.65–74.

Chandrasekaran, B. & Josephson, J.R., 2000. Function in Device Representation. *Engineering with Computers*, 16(3/4), pp.162–177.

Chen, K.-M. & Liu, R.-J., 2005. Interface Strategies in Modular Product Innovation. *Technovation*, 25(7), pp.771–782.

Clausing, D.P., 2004. Operating Window: An Engineering Measure for Robustness. *Technometrics*, 46(1), pp.25–29.

CMS, 2013. *Centers for Medicare & Medicaid Services*, Available at: https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Artifacts.html.

Cockburn, A., 2000. *Writing Effective Use Cases*.

Crawley et al., 2004, The Influence Of Architecture In Engineering Systems. Engineering Systems Monograph.

Crilly, N., 2012. Function Propagation through Nested Systems. *Design Studies*, 34(2), pp.216–242.

Daniels, J. & Bahill, T., 2004. The Hybrid Process That Combines Traditional Requirements and Use Cases. *Systems Engineering*, 7(4), pp.303–319.

Danilovic, M. & Browning, T.R., 2007. Managing Complex Product Development Projects with Design Structure Matrices and Domain Mapping Matrices. *International Journal of Project Management*, 25(3), pp.300–314.

Da Silveira, G., Borenstein, D. & Fogliatto, F.S., 2001. Mass Customization: Literature Review and Research Directions. *International Journal of Production Economics*, 72(1), pp.1–13.

Davis, TP. (2006) Science, engineering, and statistics. *Applied Stochastic Models in Business and Industry*, 22 (5-6), pp. 401-430.

Deng, Y.M., 2002. Function and Behaviour Representation in Conceptual Mechanical Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 16, pp.343–362.

Department of Defence (DoD), 2001. *Systems Engineering Fundamentals*, Defence Acquisation University Press, Fort, Belvoir.

Devanathan, S. et al., 2009. A Working Knowledge Model for Supporting Early

Design through Visual Tools. In *International Conference on Engineering Design, ICED09*. 24-27 August, Stanford, CA, USA, pp. 299–310.

Devanathan, S. & Ramani, K., 2011. Towards Enabling Visual Design Exploration involving Multiple Abstraction of Design Descriptions. In *Proceedings of IDETC/CIE*. 28-31 August, Washington DC, USA, pp. 1–15.

Dictionary.com, 2016. Dictonary. Available at: http://www.dictionary.com/.

Dong, Q., 2002. *Predicting and Managing System Interactions at Early Phase of the Product Development Process*. PhD Thesis, MIT.

Dori, D., 2002. *Object Process Methodology: A Holistic Systems Paradigm*, Springer Berlin Heidelberg.

Driessen, K. et al., 2006. The FEI Small Dual Beam Product Family. *System Research Forum*, 1, pp.73–87.

Durugbo, C., Tiwari, A. & Alcock, J.., 2011. A Review of Information Flow Diagrammatic Models for Product-Service Systems. *International Journal of Advanced Manufacturing Technology*, 52, pp.1193–1208.

Eckert, C., 2013. That Which is not Form. The Practical Challenges in Using Functional Concepts in Design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AI EDAM)*, 27(3), pp.217–231.

Eckert, C.M., Alink, T. & Albers, A., 2010. Issue Driven Analysis of an Existing Product at Different Levels of Abstraction. In *Design Methods: Proceedings of the International Design Conference*. Dubrovnik, Croatia, 17-20 May., pp. 673–682.

Eisenbart, B., 2014. Supporting Interdisciplinary System Development through Integrated Function Modelling. PhD dissertation. University of Luxembourg.

El-Haik, B.S., 2005. *Axiomatic Quality: Integrating Axiomatic Design with Six-Sigma, Reliability, and Quality Engineering.*, John Wiley & Sons, Inc.

Engels, G. et al., 2005. Process Modelling using UML. In *Process Aware Information Systems: Bridging People and Software Through Process*

Technology. pp. 1–34.

Erden, M.S. et al., 2008. A Review of Function Modeling : Approaches and Applications. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 22(02), pp.147–169.

Eriksson, M., Borg, K. & Börstler, J., 2008. Use Cases for Systems Engineering — An Approach and Empirical Evaluation. *Systems Engineering*, pp.39–60.

Estefan, J.A., 2007. Survey of Model-Based Systems Engineering (MBSE) Methodologies.

Ford, 2004. *FMEA Handbook Version 4.1.*

Ford, 1997. *Ford Technical Education Porgram: Systems Engineering Fundamental* V2.0 ed.

Fosse, E. & Delp, C.L., 2013. Systems Engineering Interfaces: A Model Based Approach. *2013 IEEE Aerospace Conference*, pp.1–8.

French, M.J., 1985. *Conceptual Design for Engineers*, London: Springer-Verlag.

Friedman, G. et al., 1998. *Representation of Constraints in the System Architecture*, MIT, Cambridge, MA.

Fritzsche, 2008. Analysis of Interfaces and Interface Management of Automobile Systems. SAE Technical Paper.

Gedell, S., Michaelis, M.T. & Johannesson, H., 2011. Integrated Model for Co-Development of Products and Productions Systems - A Systems Theory Approach. *Concurrent Engineering*, 19(2), pp.139–156.

Gero, J., 1990. Design prototypes: A Knowledge Representation Scheme for Design, 11(4): 26-36. *AI Magazine*, 11(4), pp.26–36.

Golkar, A. & Crawley, E.F., 2014. A Framework for Space Systems Architecture under Stakeholder Objectives Ambiguity. *Systems Engineering*, 17(4), pp.479–502. Available at: http://doi.wiley.com/10.1111/sys.21286.

Grady, J.O., 2006. *System Requirements Analysis*, Amsterdam: Elsevier.

Grix, J. (2004) *The foundations of research.* New York: Palgrave Macmillan.

Gzara, L., Rieu, D. & Tollenaere, M., 2003. Product Information Systems Engineering : An Approach for Building Product Models by Reuse of Patterns. *Robotics & Computer Integrated Manufacturing*, 19, pp.239–261.

Hamraz, B. et al., 2013. Change Prediction using Interface Data. *Concurrent Engineering*, 21(2), pp.141–154.

Harel, D., 1987. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, pp.231–274.

Haskins, C. et al., 1995. *NASA Systems Engineering Handbook*, NASA.

Hassan, M., Tajammul, A. & Asim, M., 2010. Application of QFD in Digital Electronics Industry. In *Proceedings of the 2010 IEEE IEEM*. pp. 482–486.

Hatley, D.. & Pirbhai, I.., 1988. *Strategies for Real Time System Specification*, Dorset House, New York.

Hauser, J.. & Clausing, D., 2009. *The House of Qaulity*, Harvard Business Review.

Helmer, R., Yassine, A. & Meier, C., 2010. Systematic Module and Interface Definition using Component. *Journal of Engineering Design*, 21(6), pp.647–675.

Henshall, E., Rutter, B. & Souch, D., 2015. Extending the Role of Interface Analysis within a Systems Engineering Approach to the Design of Robust and Reliable Automotive Product. *SAE Int. J. Mater. Manf*, 8(2), pp.322–335.

Hitchins, D., 2007. *Systems Engineering: A 21st Century Systems Methodology*, Chichester: John Wiley & Sons Ltd.

Hoffmann, H.P., 2011. *Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering* Deskbook 3., IBM Corporation.

Holley, V., Jankovic, M. & Yannou, B., 2014. Physical Interface Ontology for

Management of Conflicts and Risks In Complex Systems. *Concurrent Engineering*, 22(2), pp.148–161.

Holman, R., Kaas, H.-W. & Keeling, D., 2003. The Future of Product Development. *McKinsey Quarterly*, 3, pp.28–39.

Hood, C. et al., 2008. *Requirements management, the interfaces between requirements development and all other systems engineering processes*, Springer-Verlag, Heidelberg.

Hubka, V. & Eder, W.E., 1996. *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge.*

IEEE 1471, 2000. *Recommended practice for architectural description for software-intensive systems*, New York.

INCOSE, 2000. *Systems Engineering Handbook* 2nd ed.,

INCOSE, S.H., 2006. *Object Oriented System Engineering Method*, INCOSE SE Handbook Version 3.

INCOSE SEH Working Group, 2011. *Systems Engineering Handbook* Version 3.

INCOSE SEH Working Group, 2004. *Systems engineering handbook* version 2a.,

Jarratt, T.A.W., 2004. *A Model Based Approach to Support the Management of Engineering Change*. PhD Thesis, Cambridge University.

JLR, 2014. *JLR FMEA Handbook Version 3.0 .*

Kaufman, J.J. & Woodhead, R., 2006. *Stimulating Innovation in Products and Services*, Wiley.

King, A.M. & Sivaloganathan, S., 1998. Development of a Methodology for using Function Analysis In Flexible Design Strategies. , 212, pp.215–230.

Ko, Y.T., 2013. Optimizing Product Architecture for Complex Design. *Concurrent Engineering: Research and Applications*, 21(2), pp.87–102.

Komoto, H. & Tomiyama, T., 2011. A Theory of Decomposition In System Architecting. In *International Conference on Engineering Design*.

Kossiakoff, A. et al., 2011. *Systems Engineering Principles and Practice* Second., New Jersey: John Wiley & Sons, Inc.

Lalli, V.R., Kastner, R.E. & Hartt, H.N., 1997. Training Manual for Elements of Interface Definition and Control Training Manual for Elements of Interface Definition and Control.

Liang, V.-C. & Paredis, C.J.J., 2004. A Port Ontology for Conceptual Design of Systems. *Journal of Computing and Information Science in Engineering*, 4(3), p.206.

Lopez-Mesa, B. & Bylund, N., 2011. A Study of the Use of Concept Selection Methods from Inside a Company. *Research in Engineering Design*, 22(1), pp.7–27.

Maier, J.R.A. & Fadel, Æ.G.M., 2009. Affordance-Based Design Methods for Innovative Design , Redesign and Reverse Engineering. *Research in Engineering Design*, pp.225–239.

Maier, J.R.A. & Fadel, G.M., 2003. Affordance-Based Methods for Design. In *ASME Conference on Design Theory and Methodology*. Chicago, IL.

Malan, R. & Bredemeyer, D., 2001. Functional Requirements and Use Cases. Available at: http://www.bredemeyer.com.

Malmqvist, J., 2002. A Classification of Matrix-Based Methods for Product Modelling. In *International Design Conference*. 14-17 May, Dubrovnik, Croatia, pp. 203–210.

Martin, J.N., 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products*, New York: CRC Press.

Martin, M. V & Ishii, K., 2002. Design for Variety : Developing Standardized and Modularized Product Platform Architectures. *Research in Engineering Design*, 13, pp.213–235.

Mikkola, J.H., 2001. Modularity and Interface Management of Product Architectures. In *International Conference on Management of Engineering and Technology, PICMET'01*. Portland, pp. 599–609.

Miller, T.D. & Elgard, P., 1998. Defining Modules , Modularity and Modularization Evolution of the Concept in a Historical Perspective. In *13th IPS Research Seminar*. Aalborg University, Fuglsoe.

Muller, G., 2004. *CAFCR : A Multi-view Method for Embedded Systems Architecting ; Balancing Genericity and Specificity*. Technische Universiteit Delft.

NASA, 2007. *Systems Engineering Handbook*,

Nilsen, A.F. & Muller, G., 2014. Use Cases and Non-functional Requirements Presented in Compact System Description A3s. In *INCOSE*. Las Vegas.

Nomaguchi, Y., Taguchi, T. & Fujita, K., 2006. Proposal of Knowledge Model for Designing Product Architecture and Product Family. In *Proceedings of IJCC workshop on Digital Engineering*.

Otto, K. & Wood, K., 2001. *Product Design: Techniques in Reverse Engineering and New Product Development*, New Jersey: Prentice-Hall.

Pahl, P. et al., 2007. *Engineering Design: A Systematic Approach* 3rd Editio., Springer.

Parslov, J.F. & Mortensen, N.H., 2015. Interface Definitions in Literature : A Reality Check. *Concurrent Engineering: Research and Applications*, 23(3), pp.183–198.

Pimmler, T.U. & Eppinger, S.D., 1994. Integration Analysis of Product Decompositions. In *ASME Design Theory and Methodology Conference*.

Pugh, S., 1991. *Total Design: Integrated Methods for Successful Product Engineering*, Harlow: Addison-Wesley.

Rahim, A.R.. & Baksh, M.S.., 2003. Application of Quality Function Deployment (QFD) Method for Pultrusion Machine Design Planning. *Industrial*

Management & Data Systems, 103(6), pp.373–387.

Rahmani, K., 2012. *Computerized Interface Control*. PhD Thesis, McGill University.

Rahmani, K. & Thomson, V., 2012. Ontology Based Interface Design and Control Methodology for Collaborative Product Development. *Computer-Aided Design*, 44(5), pp.432–444.

Rational Software, 2003. *Rational Unified Process for Systems Engineering*, IBM Ratioanl Software.

Reverso Dictionary, 2016. Reverso. Available at: http://dictionary.reverso.net/.

Rosenman, M. & Gero, J., 1998. Purpose and Function in Design: From the Socio-Cultural to the Techno-Physical. *Design Studies*, 19, pp.161–186.

Sage, A.P. & Lynch, C.L., 1998. Systems integration and Architecting: An Overview of Principles, Practices, and Perspectives. *Systems Engineering*, 1(3), pp.176–227. Available at:

Salado, A. & Nilchiani, R., 2014. A Categorization Model of Requirements Based on Max-Neef ’s Model of Human Needs. *Systems Engineering*, 17(3), pp.348–360.

Scalice, K.R., Andrade, L.F.S. & Forcellini, F.A., 2008. A Design Methodology for Module Interfaces. In *Collaborative Product and Service Life Cycle Management for a Sustainable World*. pp. 297–304.

Schlatt, H., 2004. The Aircraft / Store Common Interface. *SAE Technical Paper*.

Seaman, C., 1999. C. Seaman, Qualitative methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software Engineering*, 25(4), pp.557–572.

Sellgren, U., 2006. A Model-Based Approach To Situated Design. In *First Nordic Conference on Product Lifecycle Management*. Goteborg, pp. 51–62.

Sellgren, U., 1998. Attachment of Mating Faces - An Interrelational Feature

Approach. In *8th Int. ANSYS Conference*. Pittsburgh, PA, pp. 95–110.

Sellgren, U. & Anderson, S., 2005. The Concept of Functional Surfaces as Carriers of Interactive Properties. In *Int. Conf. on Engieering Design (ICED)*. Melbourne, pp. 1–15.

SGTS, 2012. Introduction to Systems. Available at: http://www.educationscotland.gov.uk/.

Soderborg, N.R., Crawley, E.F. & Dori, D., 2002. System Definition for Axiomatic Design Aided by Object-Process Methdology. In *Proceedings of ICAD*. June 10&11, Cambridge, MA, pp. 1–7.

Sosa, M.E., Eppinger, S.D. & Rowles, C.M., 2003. Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions. *Journal of Mechanical Design*, 125(2), pp.240–252.

Steward, D.V., 1981. The Design Structure System : A Method for Managing the Design of Complex Systems. *IEEE Transactions on Engineering Management*, 28(3), pp.71–74.

Stone, R.B. & Wood, K.L., 2000. Development of a Functional Basis for Design. *Journal of Mechanical Design*, 122(4), pp.359–370.

Suh, N.P., 1998. Axiomatic Design Theory for Systems. *Research in Engineering Design*, pp.189–209.

Suh, N.P., 2001. *Axiomatic Design: Advances and Applications*, Oxford University Press.

Tang, D. et al., 2010. Product Design Knowledge Management based on Design Structure Matrix. *Advanced Engineering Informatics*, 24(2), pp.159–166.

Tapke, J. et al., 2014. IE361: House of quality: Steps in Understanding the House of Quality.

Tate, D., 1999. *A Roadmap for Decomposition: Activities, Theories, and Tools for System Design*. Massachusetts Institute of Technology.

Tomiyama, T., 2012. Architecture-Centric Model-Based Product Development. In *Mechatronics*. Linz, Austria, pp. 434–443.

Tomiyama, T. et al., 2009. Design Methodologies: Industrial and Educational Applications. *CIRP Annals - Manufacturing Technology*, 58, pp.534–565.

Uddin, A. et al., 2016. A Framework for Complex Product Architecture Analysis using an Integrated Approach. *Concurrent Engineering: Research and Applications*, 24(3), pp.195-210.

Ullman, D.G., 2010. *The Mechanical Design Process* 4th ed., Singapore: McGraW Hill.

Ulrich, K., 1995. The Role of Product Architecture in the Manufacturing Firm. *Research Policy*, 24(3), pp.419–440. Available at: http://linkinghub.elsevier.com/retrieve/pii/0048733394007753.

Ulrich, K.T. & Eppinger, S.D., 2003. *Product Design and Development* 3rd ed., New York: McGraw-Hill/Irwin.

Umeda, Y. et al., 1990. Function, Behaviour, and Structure. In *Applications of Artificial Intelligence in Engineering Design*. Berlin: Springer-Verlag, pp. 177–193.

Umeda Y. et al., 1996. Supporting Conceptual Design based on the Function-Behaviour-State Modeler. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 10(4), pp.275–288.

Vermaas, P.., 2009. The Flexible Meaning of Function in Engineering. In *International Conference on Engineering Design*. pp. 113–124.

Vermaas, P.E., 2010. Technical Functions: Towards Accepting Different Engineering Meanings with the Overall Account. In *Proceedings of TMCE2010 Symposium*. Ancona, Italy, pp. 183–194.

Warell, A., 2001. *Design syntactics: A Functional Approach to Visual Product Form*. The Chalmers University of Technology, Gothenburg, Sweden.

Wasson, C.S., 2006. *System Analysis, Design, and Development: Concepts ,*

Principles , and Practices, New Jersey: John Wiley & Sons, Inc.

Webb, R.D., 2002. *Investigation into the Application of Robustness and Reliability Tools to the Design Process*. University of Bradford.

Weilkiens, T., 2007. *Systems Engineering with SysML/UML*, Morgan Kaufmann OMG Press.

Wie, M.J. Van et al., 2001. Interfaces and Product Architecture. In *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. Sept 9-12, Pittsburgh, Pennsylvania, pp. 1–12.

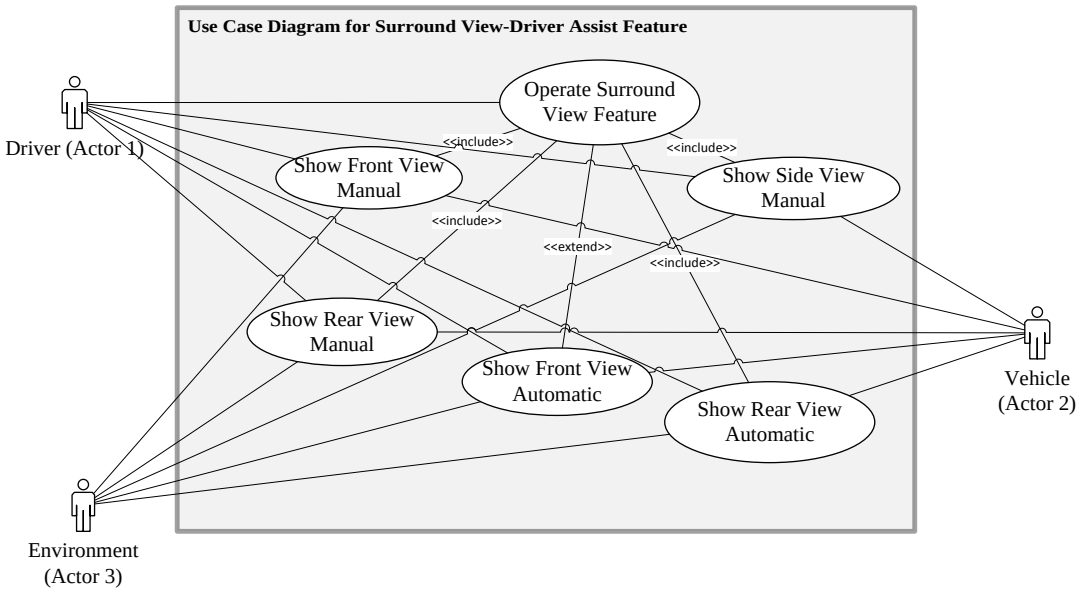
Yildirim, U. & Campean, I.F., 2014. Development of a Structured Approach for Decomposition of Complex Systems on a Functional Basis. In *27th International Conference on CADCAM, Robotics and Factories of the Future*. pp. 1–8.

Zheng, X. & Pulli, P., 2007. Improving Mobile Services Design: A QFD Appproach. *Computing and Informatics*, 26, pp.369–381.

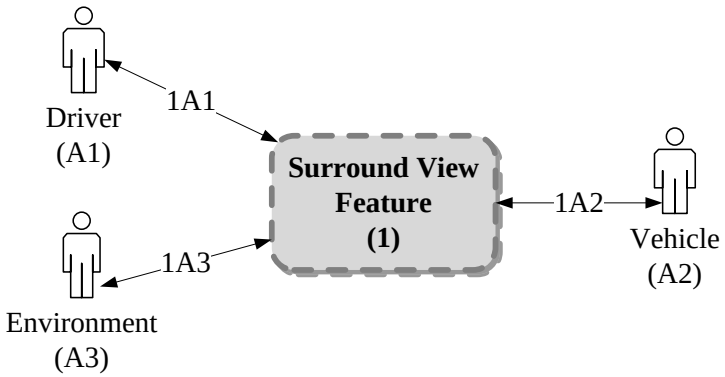
Zou, J. & Du, Q., 2013. A Functional Reasoning Cube Model for Conceptual Design of Mechatronic Systems. *Strojniški vestnik – Journal of Mechanical Engineering*, 59(5), pp.323–332.

Appendices

Appendix A: Feature level: Surround view-driver assist feature



Use case diagram



Context diagram

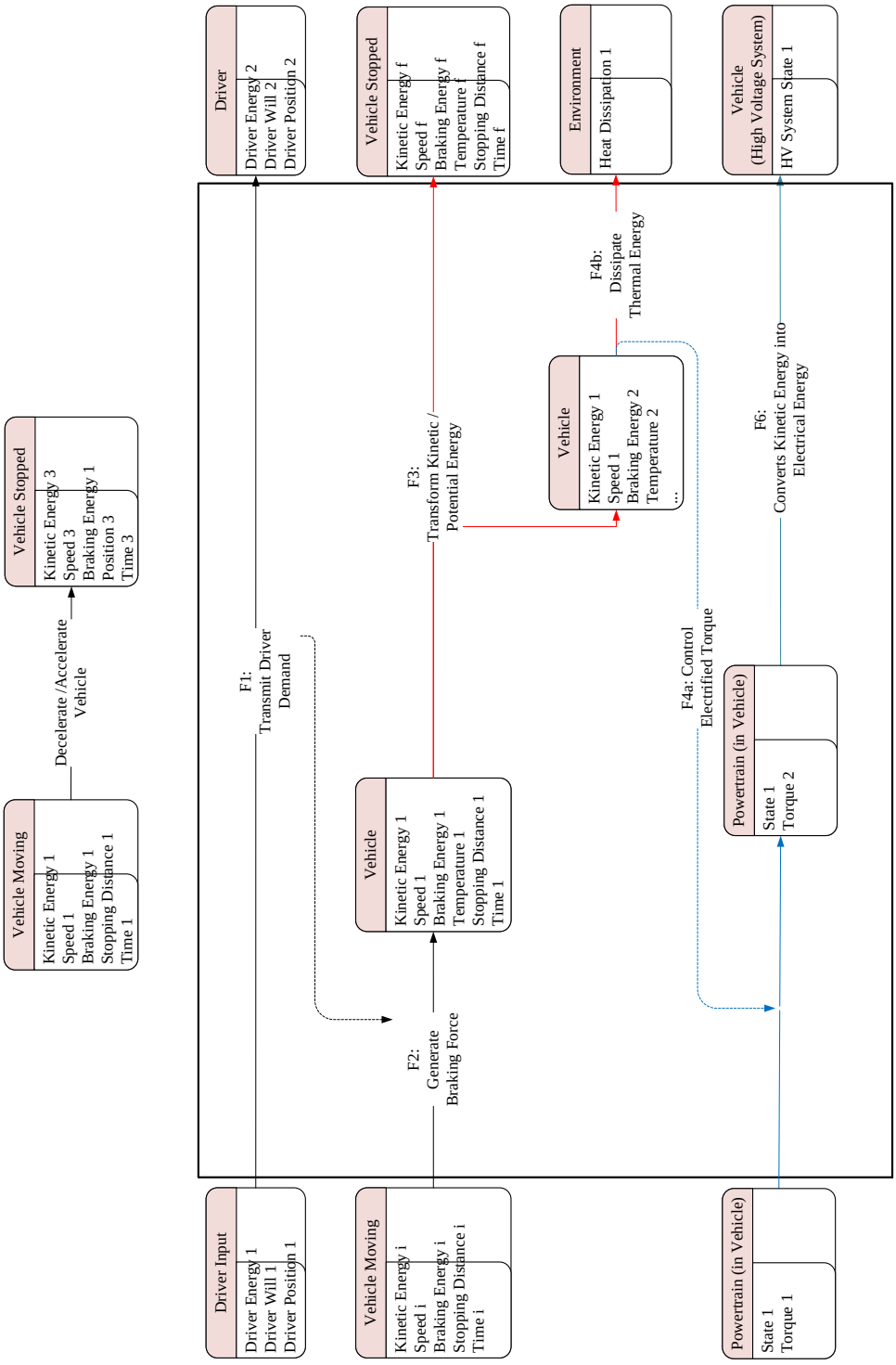
S2: Interfac	S3: Interaction Operations		S4: Interaction Exchange						S5: Exchange Effect		S6: System of Interest Requirements					S1: Use Cases	
C1	C2	C3	C4	C5	C6	C7	C8		C9	C10	C11				C12	C13	
Feature Actors	Steps	Description of Operations	Exc. Type	Exchange Description	Exchange Properties	From	To		Attribute Type	Criticality	Functional Requirement Object/Noun Verb	Input	Output		Non-functional performance requirement	Show Front View Manual	Show Rear View Manual
Main Success Scenario (Desired Interactions)																	
SV Feature (1) - Driver (A1)	1A11	Use case begins: Driver activates / requests front view	E	Driver Input	Driver Energy (J)	Driver	JVF	A03: Accomodation & Usage		2	Accept	Driver Input for activation			No. of Activation points (Min driver interface = 1)	X	
			I	ON Activation Signal	Signal Frequency (Hz)	Driver	JVF	A04: HMI & Audio-Visual Performance		2	Sense	ON Activation Signal			Min and Max. effort for activation (X < N < Y)	X	
	1A12	Driver deactivates the front view	E	Driver Input	Driver Energy (J)	Driver	JVF	A03: Accomodation & Usage		2	Accept	Driver Input			Min and Max response time (1 < sec < 0.5)	X	
			I	Activation Signal Off	Signal Frequency (Hz)	Driver	JVF	A04: HMI & Audio-Visual Performance		2	Sense	OFF Activation Signal			Min and Max. impact for activation (X < N < Y)	X	
	1A13	SV feature functions over a long period usage by driver	I	Usage Cycle		Driver	JVF	A17: Durability		1	Maintain	Operational Life			Min and Max response time (1 < sec < 0.5)	X	
	1A14	Feature informs Driver about front junction view	I	Front View Image	Image Angle (180 deg) Image Size Image Format Image Depth Field Image Transparency Image Duration	JVF	Driver	A04: HMI & Audio-Visual Performance		1	Process	Front View Image			Operational Usage (X < No. of Cycles < Y)	X	X
														Display	Image Color (Black & White < color < Real)	X	
														Front View Image symmetrically	View Angle (180 deg)	X	
	1A15	SV Feature stops functioning: use case ends	I	No information flow		JVF	Driver	A04: HMI & Audio-Visual Performance		0	Stop			FV Information		X	
Exceptional Scenario (Undersired																	
	1A16	Driver detects the front view image is not clear	I	Unclear Image	Blurred Image	JVF	Driver	A04: Vehicle HMI & Audio-Visual Performance		-1	Adjust	Unclear Image			Image Sharpness Image Resolution	X	
Main Success Scenario (Desired Interactions)																	
SV Feature (1) - Vehicle (A2)	1A21	Vehicle process the SV feature by validating user request	I	ON Activation Signal	Activation Signal Frequency (Hz)	Vehicle	JVF	A04: Vehicle HMI &		1	Verify	ON Activation Signal			Activation Frequency (Hz)	X	X
															Min and Max response time (1 < sec < 0.5)	X	X
			E	Activation Energy	Electrid Energy (mW)	Vehicle	JVF	A04: Vehicle HMI &		2	Require	Energy for activation			Min and Max power consumption (1 < watt <0.5)	X	X
Exceptional Scenario / Undersired																	
	1A22	Front view image vibrates due to vehicle motion on rough roads	E	Vibrational Energy	Vibrations (J)	Vehicle	JVF	A04: Vehicle HMI & Audio-Visual Performance		-1	Sustain	Vehicle Unstability				X	X
											Avoid/Resist		Image unstability			X	X
Main Success Scenario / Desired Interactions																	
SV Feature (1) - Environm			I	Stationary Object View	St. Object Size (mm2) St. Object Distance (mm)	Env.	JVF				Capture	Stationary Object Image			St. Object Size (mm2)	X	X
											Evaluate		Stationary Object Image		Distance of St. Object (X<m<Y)	X	X
					Moving Object Speed						Capture	Moving Objects Image			Moving Object Size (mm2)	X	X

Requirements derivation via IAT tool

Appendix B: System level: Regenerative braking system

Interface	Specification of Interaction Scenarios						Exchange Effect / Impact	System of Interest Requirements					System Use Cases			
								Functional Requirement				Nonfunctional Requirement (Performance related)	UC1	UC2	UC3	UC4
	Interaction Operation Description	Exc. Type	Exchange Description	From	To	Exchange Properties	Attribute Type	Verb	Object/Non-Object Input	Object/Non-Object Output			Stop Vehicle	Slow Vehicle	Restore Braking Energy	Generate Braking Energy
Braking System - User	101 the driver operates the braking system and gets feedback	E	Foot / Pedal force	braking system	Driver	N	A06 - brakes - pedal feel	Deliver		force		* Attribute: Force / stroke law (N / mm) Force / deceleration law (N / m/s²) * Target value XX +/- YY * Repeatability	X	X		
		P	Foot / Pedal stroke	Driver	braking system	mm	A06 - brakes - pedal feel	Receive	stroke				X	X		
	103 The driver perceives deceleration of the vehicle	E	Vehicle deceleration	Vehicle	driver	m/s2	A06 - brakes - pedal feel	Deliver		deceleration			X	X		
Braking System - Environment	201 the braking system dissipates heat in the environment	E	Heat dissipation	braking system	Environment	KJ or KWh	A06 - brakes - performance	Minimise		heat dissipation		KJ (target = energy to be dissipated to reduce 302a target)	X		X	
	202 The environment should allow to cool down the braking	M	Air flow	Environment	braking system	dm3 / s	A06 - brakes - performance	Import	Air			air flow or cooling coefficient resulting from aero design. Target = maximize 201				
Braking System - Vehicle	301 The braking system generates a torque on the wheels	E	Friction torque	braking system	wheel	Nm	A06 - brakes - performance / pedal feel	Transmit		Torque		Nm (target = torque needed to reach max deceleration)	X		X	X
		E	Electric torque	braking system	wheel	Nm	Energy management (driving range / CO2)	Transmit		torque		Nm (target = torque needed to reach target energy recovery on conditions given by cycles => depends on 302b)		X	X	X
	302 The system receives and sustains energy from the vehicle	E	Kinetic energy	Vehicle	Braking system	KJ or KWh	A06 - brakes - performance	Receive	energy			KJ (target = maximum energy to hold in worst case conditions, minus energy dissipated)	X		X	
		E	Kinetic energy	Vehicle	Braking system	KJ or KWh	Energy management (driving range / CO2)	Sustain	Kinetic energy			KWh / km (target = energy needed to meet driving range / CO2 vehicle target x system efficiency => depends on 303)		X	X	
	304 The Electrified Powertrain System informs the brake system of its capacity to apply electric torque	I	Electric torque capability information	Electrified Powertrain	Braking system	Torque value	Energy management (driving range / CO2)	Import	Signal			Nm (target = - torque needed to reach target energy recovery on conditions given by cycles - integrity guaranteed)				X
	306 The Braking system requests		Electric braking	Braking	Electrified		Energy management					Nm (target =				

Requirements derivation via IAT



Functional architecture via system state flow diagram

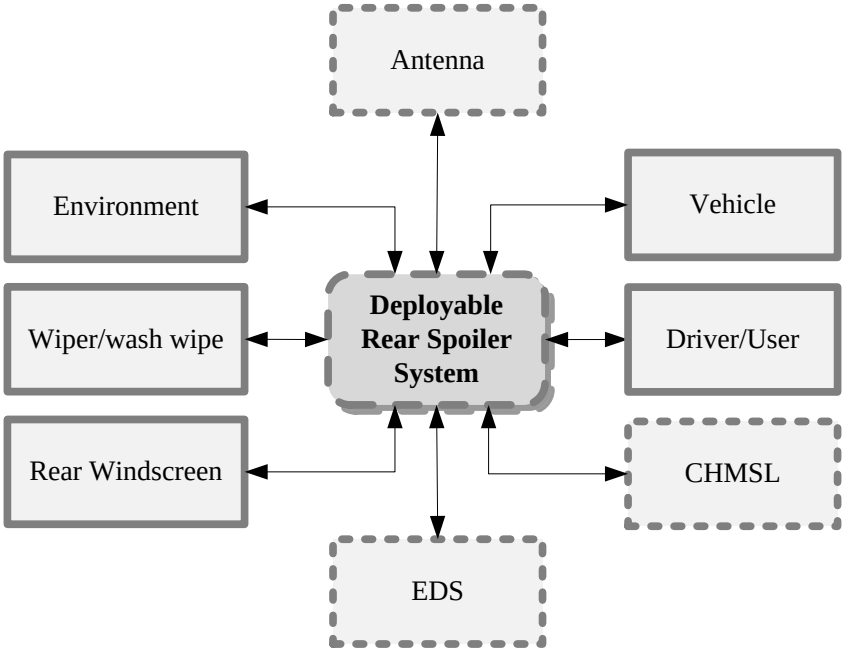
Architecture analysis via CM framework

Appendix C: Subsystem level: Electrified powertrain system's subsystems

S1: Interface	S2: Specifications of Interaction Scenarios		S3:Specification of Exchanges					S4: Exchange Effect / Impact		S5: Interface Requirements between two interacting actors				S6: System Context
C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11			C12	C13
Interface	Interaction Step	Interaction Description	Exchange Type	Exchange Description	Exchange Properties	From	To	Attribute Type or Purpose	Critical	Functional Requirement Verb	Object Input	Object Output	Nonfunctional Requirement (Performance Related)	Charge Battery
Battery Pack (BP) (A) - Charger (C) (B) Interface	AB-1 C sided	Charger charges the battery pack	E	Electric energy	High voltage/high current (V)	C	BP	A11: Energy Management	2	Transmit		Electrical power	High voltage (72 +/- 2 Vdc)	X
													Electrical Power (X < Watt < Y)	X
	AB-1 BP sided	Battery pack is charged by charger								Receive	Electrical Power		High voltage (72 +/- 2 Vdc)	X
													Electrical Power (X < Watt < Y)	X
	AB-2	State of battery's information to charger	I	Battery thermal state information	Temperature (degree C)	BP	C	A11: Energy Management	2	Detect		Battery state of charge	Energy consumption for temp. signal (2 +/- 0.5 mV)	X
			I	Battery state of charged information	State of charge signal (mV)	BP	C	A11: Energy Management	2	Transmit		State of charge signal	State of charge (2 +/- 0.5 mV)	X
	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Driver (D) (E1) Charger (C) (B) Interface	E1C-1	Electric shock to driver while charging	E	Electric shock	Electric flow (mV)	C	D	A11: Energy Management	-2	Isolate		Driver electrically		X
	E1C-2	Driver is informed about the mains power transmission	I	Visual information	Luminous flux (lumens)		D	A9: HMI & Audio Visual Performance	1	Display		Visual information regarding		X
	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Hardware - hardware interface analysis: Battery pack - charger interface

Appendix D: Subsystem level: Deployable active rear spoiler



Context Diagram

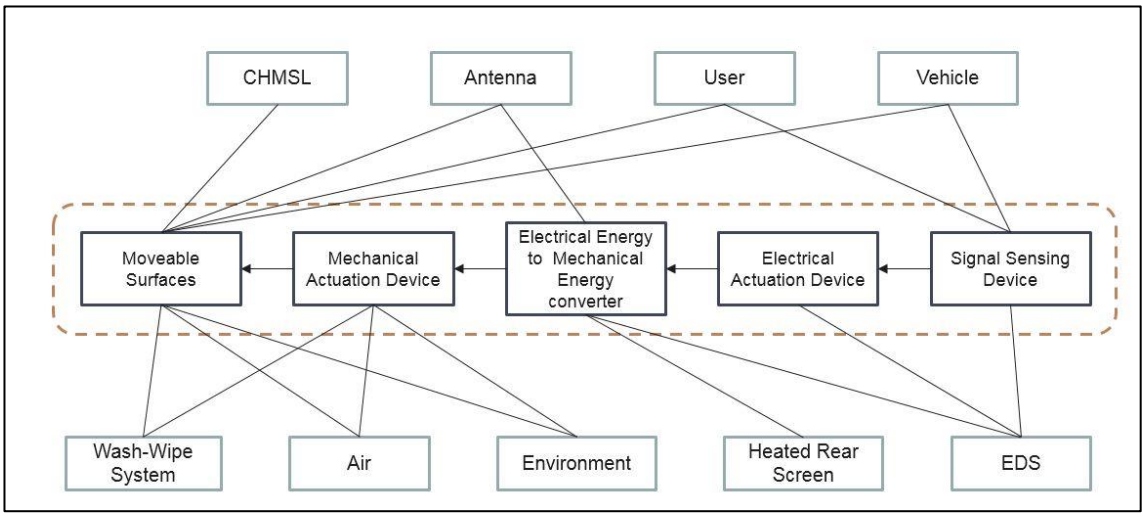
Interface	Description of Interaction Operations and Exchanges						Exchange Effect	System of Interest Requirement and Specification				Use Cases		
								Functional Requirement						
								Verb	Object/Noun		Requirement Specifications (Attribute + Target Value +	Operate the DRS	Showroom mode	Eco mode (Low drag mode)	Dynamic Mode (>100mph)
Interface	Description of Interface	Exchange Type (P, M, E, I)	Exchanges: Object/ Noun Ouput	From	To	Exchange Specification (Attribute + Target Value +	Purpose (Attributes)		Input	Output					
CHMSL (Centre high mounted stop light) - Active Spoiler	CHMSL Located in the spoiler	P	Enclosed space	CHMSL	Spoiler	Distance (mm)	A01: Safety A02: PQ	Install	CHMSL inside spoiler		Installation space x< mm <y	x	x	x	x
	CHMSL (structurally) supported by the spoiler	E	Weight supported (force)	CHMSL	Spoiler	Force (N)	A09: Durab. & Reliab	Sustain	CHMSL Load		weight force x< kg <y	x	x	x	x
	Visibility of CHMSL (for other vehicles/ legal req) when spoiler moves	P	Position and angle of CHMSL	CHMSL	User/ Other vehicles	Position (mm, deg)	A01: Safety A03: A&U	Visible	CHMSL Light emitted		Visible at viewing angles x< deg <y	x	x	x	x
Wiper (& Wiper spindle) - Active Spoiler	Wiper mechanism requires clearance of 10 mm between any moving part	P	Clearance distances	Spoiler	Wiper	Distance (mm)	A01: Safety A14: AWC&V	Maintain	Clearance from wiper		Clearance x< mm <y	x	x	x	x
	Access required to wiper when replacing wiper blade	P	Clearance distances	Spoiler	Wiper	Distance (mm)	A13: Service & Ownership A17: Total cost of ownership	Access	Wiper Blade		Clearance x< mm <y		x		
	Water and cleaning fluid splashing onto spoiler	M	Chemicals splashing	Wiper	Spoiler	Amount of liquid (ml)	A01: Safety A09: Durab. & Reliab. A14: AWC&V	Resist	Water and Chemicals		amount of chemicals x< ml <y	x	x	x	x
Antenna - Active Spoiler	Requires 300mm clearance to antenna	P	Clearance distances	Spoiler	Antenna	Distance (mm)	A04: HMI & AV Performance A08: NVH & Sound Qual.	Maintain	Clearance to antenna		Clearance x< mm <y	x	x	x	x
EDS (Electrical Distribution System) - Active Spoiler	Harnesses located in the spoiler	P	Enclosed space	Harnesses	Spoiler	Distance (mm)	A12: Energy & Energy Management	Install	Harness inside spoiler		Installation space x< mm <y	x	x	x	x
	Harnesses supported by the spoiler	E	Weight of harnesses in spoiler	Harnesses	Spoiler	Force (N)	A09: Durab. & Reliab	Sustain	Harness mass		weight force x< kg <y	x	x	x	x
Heated Rear screen - Active Spoiler	Access to rear screen to fix or replace required	P	Limited space	Spoiler	Rear screen	Distance (mm)	A13: Service & Ownership A17: Total cost of ownership	Maintain	Clearance to screen		Clearance x< mm <y		x		

Requirements derivation via IAT

Interface	Description of interaction and Exchange							Exchange Effect	System of Interest Requirement and Specification															
			Exchanges:						Functional Requirement															
Interface	Description of Interface	Exchange Type (P, M, E, I)	Object/ Noun	From	To	Exchange Specification (Attribute + Target Value + Relation+ Unit)	Verb	Purpose (Attributes)	Object/Noun		Requirement Specifications (Attribute + Target Value + Relation+ Unit)	F1 Actuate EE	F2 Convert vehicle speed into	F3 Convert UI into control signal	F4 Convert EE into ME	Transmit ME	F5 Deploy Spoiler (MK)	F6 Retract Spoiler (MK)
			Ouput						Input	Output														
CHMSL (Centre high mounted stop light) - Active Spoiler	CHMSL Located in the spoiler	P	Enclosed space	CHMSL	Spoiler	Distance (mm)	Install	A01: Safety A02: PQ	CHMSL inside spoiler		Installation space xx mm cy						X	X
	CHMSL (structurally) supported by the spoiler	E	Weight supported (force)	CHMSL	Spoiler	Force (N)	Sustain	A09: Durab. & Reliab	CHMSL Load		weight force xx kg cy						X	X
	Visibility of CHMSL (for other vehicles' legal req) when spoiler moves	P	Position and angle of CHMSL	CHMSL	User Other	Position (mm, deg)	Visible	A01: Safety A03: A&U	CHMSL Light emitted		Visible at viewing angles xx deg cy						X	X
Wiper (& Wiper spindle) - Active Spoiler	Wiper mechanism requires clearance of 10 mm between any moving part	P	Clearance distances	Spoiler	Wiper	Distance (mm)	Maintain	A01: Safety A14: AVC&V A15: Service &	Clearance from wiper		Clearance xx mm cy						X	X
	Access required to wiper when replacing wiper blade	P	Clearance distances	Spoiler	Wiper	Distance (mm)	Access	Ownership	Wiper Blade		Clearance xx mm cy							X
	Water and cleaning fluid splashing onto spoiler	M	Chemicals splashing	Wiper	Spoiler	Amount of liquid (ml)	Resist	A01: Safety A03: Durab. & Reliab.	Water and Chemicals		amount of chemicals xx ml < y					X		
User - Active Spoiler	Spoiler may be overloaded e.g. user leans on it when loading roof box	E	Human weight (force)	User	Spoiler	Force (N)	Resist	A03: Durab & Reliab A11: Total cost of	User force (e.g. weight)		force xx N cY						X	X
	Cleaning the vehicle using a jet washer	E	Water pressure (force)	Jet washer	Spoiler	Force (Pa)	Resist	A03: Durab & Reliab A13: Service &	Water pressure		pressure force xx N cy						X	X
	Cleaning the vehicle using an automatic car wash	E	Load on spoiler (force)	Car wash	Spoiler	Force (N)	Resist	A03: Durab & Reliab A13: Service &	Car wash brush forces		force xx N cy						X	X
	Cleaning using chemicals	E	Chemical abrasion/ degradation	Chemicals	Spoiler	Chemical energy (J)	Resist	A03: Durab & Reliab A13: Service &	Car cleaning chemicals		chemical energy xx J cy						X	X
	Cleaning the vehicle using chemicals/ water	M	Material deposited on the spoiler	Chemicals/ water	Spoiler	Amount of chemicals/ water (ml)	Resist	A03: Durab & Reliab A13: Service &	Car cleaning chemical deposits		amount of chemicals xx ml < u						X	X
	User's rear vision obstructed when driving bu the spoiler	P	Position of spoiler	Spoiler	User	Distances (mm)	Maintain	A01: Safety A03: A&U	Rear visibility		Vision angle/ vision area (deg, mm^2, mm, etc.)						X	
	Approved operator replacing/ fixing spoiler with correct tools	E	Force applied to fixings	User	Spoiler	Force (N)	Yield	A03: Durab & Reliab A13: Service &	User force	Spoiler fixings	Yield to force xx N						X	X
Vehicle - Active Spoiler	Movement of Spoiler at certain speed range	E	Mechanical Energy	Vehicle	Spoiler	Mechanical Speed (m/sec)	Deliver	A03: Durab & Reliab A14: AVC&V	Mechanical energy		Movement Speed (m/sec)		X		X			
	Lock out required to hold spoiler in position	E	Force (N)	Vehicle	Spoiler	Force (N)	Resist	A03: Durab & Reliab	Forces of spoiler movement		Resist force xx N cY				X	X	X	X
	Spoiler should deploy at a set speed / set conditions	I	Information signal	Vehicle	Spoiler	Electrical pulse (V)	Receive	A05: Vehicle Dynamics A07: Product Performance	Electrical signal		Response Time x < sec < Y		X		X	X	X	
	Spoiler should retract at a set speed / set conditions	I	Information signal	Vehicle	Spoiler	Electrical pulse (V)	Receive	A00: Product Efficiency	Electrical signal		electrical signal xx V cY	X	X	X		X		X
	Spoiler should rapond at specific vehicle speed	I	Information signal	Vehicle	Spoiler	Vehicle Speed information (m/s)	Detect	A05: Vehicle Dynamics	Vehicle Speed		Vehicle Speed (X<m/sec<Y)		X		X		X	
	Spoiler located at the top of the tailgate	P	Packaging	Spoiler	Vehicle	Distance (mm)	Install	A14: AVC&V A15: TASE	To top of tailgate		spoiler in position: X, Y, Z mm away from tailgate.							
	Spoiler must be interchangable with the static spoiler option	P	Spoiler fixings and mountings	Spoiler	Vehicle	Position (mm)	Install	A11: Weight A13: Service &	Both spoilers can be installed in same location		spoiler in position: X, Y, Z mm away from tailgate.					X	X	

Use Cases
UC4: Provide Aesthetic Look
UC3: Reduce Rear Sailing
UC2: Reduce Rear Lift
UC1: Provide Low Drag

Physical Architecture
Electrical Actuation Device
Signal Sensing Device
Electro-Mechanical Converter
Mechanical Actuation Device



System boundary diagram